IT-Sicherheit

Konzepte – Verfahren – Protokolle

von
Prof. Dr. Claudia Eckert

8., aktualisierte und korrigierte Auflage

Oldenbourg Verlag München

Prof. Dr. Claudia Eckert ist seit 2009 Inhaberin des Lehrstuhls für Sicherheit in der Informatik an der TU München. Gleichzeitig ist sie Direktorin der Fraunhofer Einrichtung für Angewandte und Integrierte Sicherheit (AISEC) mit Sitz in Garching bei München. Als Mitglied verschiedener nationaler und internationaler wissenschaftlicher Gremien berät Frau Prof. Eckert Unternehmen, Wirtschaftsverbände sowie die öffentliche Hand in allen Fragen der IT-Sicherheit.

Bibliografische Information der Deutschen Nationalbibliothek

Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de abrufbar.

© 2013 Oldenbourg Wissenschaftsverlag GmbH
Rosenheimer Straße 143, D-81671 München
Telefon: (089) 45051-0
www.oldenbourg-verlag.de

Das Werk einschließlich aller Abbildungen ist urheberrechtlich geschützt. Jede Verwertung außerhalb der Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Verlages unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen und die Einspeicherung und Bearbeitung in elektronischen Systemen.

Lektorat: Johannes Breimeier
Herstellung: Tina Bonertz
Einbandgestaltung: hauser lacour
Gesamtherstellung: Kösel, Krugzell

Dieses Papier ist alterungsbeständig nach DIN/ISO 9706.

ISBN 978-3-486-72138-6
eISBN 978-3-486-73587-1

Vorwort

Informationstechnologie (IT) ist heute in nahezu allen Bereichen von zentraler Bedeutung: Gesundheit, Mobilität, Bildung, Unterhaltung, Produktion, Logistik, aber auch Handel, Finanzen, und öffentliche Verwaltung. IT ist eine Schlüsseltechnologie, die neue Anwendungen und auch neue Geschäftsmodelle ermöglicht. Durch die zunehmende Einbettung von IT in Alltagsgegenstände und durch die zunehmende Dienste-Orientierung entsteht das Internet der Dinge und der Dienste. Der IT-Sicherheit kommt bei dieser Entwicklung eine Schlüsselrolle zu. Lösungen der IT-Sicherheit haben zum eine Wegbereiter-Funktion, da neue Anwendungen häufig nur eingesetzt und akzeptiert werden, wenn die Sicherheit der Daten gewährleistet wird. Zum anderen hat die IT-Sicherheit natürlich die bekannte Schutzfunktion. Gezielt und korrekt eingesetzte Maßnahmen der IT-Sicherheit reduzieren die Risiken wirtschaftlicher Schäden, die zum Beispiel durch eine unautorisierte Weitergabe von Daten oder durch kriminelle Aktivitäten wie Wirtschaftsspionage entstehen können. Maßnahmen der IT-Sicherheit sind aber auch notwendig, um vor Schäden an Leib und Leben zu schützen, die zum Beispiel durch manipulierte Gesundheitsdaten oder durch manipulierte Fahrzeugsensorik entstehen können.

Ergebnisse von Sicherheitsanalysen und die immer häufiger auftretenden Angriffe auf vernetzte IT-Systeme verdeutlichen die großen Sicherheitsprobleme der heute im Einsatz befindlichen Systeme. Zu den Hauptursachen für Sicherheitsprobleme gehören konzeptuelle Mängel, Implementierungsfehler, Fehler bei der Systemadministration sowie Benutzungsfehler, die häufig auf Unkenntnis und Unachtsamkeit zurückzuführen sind.

Das vorliegende Buch hat zum Ziel, fundierte Kenntnisse über wirksame Maßnahmen zu vermitteln, die erforderlich sind, um die vielfältigen Sicherheitsbedrohungen abzuwehren, denen heutige und zukünftige IT-Systeme ausgesetzt sind. Die Qualität eines sicheren IT-Systems hängt wesentlich davon ab, dass seine Konstruktion methodisch und systematisch erfolgt. Das Buch führt die hierfür notwendigen Techniken und Vorgehensweisen ein, wie Bedrohungsanalyse, Modellierung und Bewertung. Die zur Umsetzung der Sicherheitsanforderungen benötigten Mechanismen, Verfahren und Protokolle werden eingehend erklärt sowie anhand von Fallbeispielen erläutert.

Im Vordergrund steht dabei, die Ursachen für Problembereiche heutiger IT-Systeme zu verdeutlichen und die grundlegenden Sicherheitskonzepte mit ihren jeweiligen Vor- und Nachteilen vorzustellen. Der Leser soll ein Verständnis für die vielschichtigen Probleme sicherer Systeme erlangen sowie ein breites und grundlegendes Wissen zu deren Behebung erwerben.

Das Buch beschäftigt sich mit Fragestellungen der Sicherheit technischer Systeme im Sinne des englischen Begriffs Security. Zur Gewährleistung von Security-Eigenschaften setzt man Konzepte und Maßnahmen ein, um Bedrohungen abzuwehren, die im Wesentlichen durch absichtliche oder unabsichtliche Angriffe von außen auf das IT-System entstehen. Sicherheitsaspekte, die organisatorische Fragen betreffen, und solche, die durch den Begriff Safety charakterisiert werden, liegen außerhalb des behandelten Themenrahmens. Safety beschreibt die Funktionssicherheit eines Systems. Zu deren Gewährleistung benötigt man Konzepte und Verfahren, um Bedrohungen abzuwehren, die im Wesentlichen durch das Fehlverhalten des IT-Systems selber entstehen. Es handelt sich hierbei um Fragestellungen der Fehlervermeidung und Fehlertoleranz sowie der Steigerung der Zuverlässigkeit und Verfügbarkeit von IT-Systemen.

Das Buch richtet sich an Studierende der Informatik, Mathematik und Elektrotechnik sowie an interessierte Leser mit Informatik-Hintergrundwissen, die grundlegende Kenntnisse auf dem Gebiet der IT-Sicherheit erwerben bzw. bereits vorhandene Kenntnisse vertiefen wollen. Das Buch ist für die eigenständige Beschäftigung mit dem Stoff geeignet.

Vorwort zur achten Auflage

Das Gebiet der IT-Sicherheit ist einem sehr dynamischen Wandel unterworfen, so dass eine regelmäßige Aktualisierung notwendig ist. Die vorliegende Auflage enthält deshalb eine korrigierte und aktualisierte Überarbeitung der siebten Auflage.
Bedanken möchte ich mich ganz herzlich bei allen Lesern, die mir Hinweise auf Fehler und Unklarheiten zukommen ließen. Dank gebührt auch meinem Lektor Herrn Johannes Breimeier vom Oldenbourg-Verlag für seine Unterstützung.

Für Hinweise auf Fehler und für Verbesserungsvorschläge bin ich jederzeit dankbar. Bitte senden Sie diese an claudia.eckert@aisec.fraunhofer.de.

München, im Januar 2013
Claudia Eckert

Inhaltsverzeichnis

1	**Einführung**	**1**
1.1	Grundlegende Begriffe	3
1.2	Schutzziele	7
1.3	Schwachstellen, Bedrohungen, Angriffe	16
1.3.1	Bedrohungen	16
1.3.2	Angriffs- und Angreifer-Typen	19
1.3.3	Rechtliche Rahmenbedingungen	28
1.4	Computer Forensik	32
1.5	Sicherheitsrichtlinie	34
1.6	Sicherheitsinfrastruktur	37
2	**Spezielle Bedrohungen**	**45**
2.1	Einführung	45
2.2	Buffer-Overflow	47
2.2.1	Einführung	48
2.2.2	Angriffe	50
2.2.3	Gegenmaßnahmen	53
2.3	Computerviren	56
2.3.1	Eigenschaften	56
2.3.2	Viren-Typen	58
2.3.3	Gegenmaßnahmen	65
2.4	Würmer	68
2.5	Trojanisches Pferd	73
2.5.1	Eigenschaften	73
2.5.2	Gegenmaßnahmen	75
2.6	Bot-Netze und Spam	77
2.6.1	Bot-Netze	78
2.6.2	Spam	80
2.7	Mobiler Code	82
2.7.1	Eigenschaften	82
2.7.2	Sicherheitsbedrohungen	83
2.7.3	Gegenmaßnahmen	85
2.7.4	Mobile Apps	88

3	**Internet-(Un)Sicherheit**	**95**
3.1	Einführung	95
3.2	Internet-Protokollfamilie	97
3.2.1	ISO/OSI-Referenzmodell	97
3.2.2	Das TCP/IP-Referenzmodell	104
3.2.3	Das Internet-Protokoll IP	106
3.2.4	Das Transmission Control Protokoll TCP	111
3.2.5	Das User Datagram Protocol UDP	114
3.2.6	DHCP und NAT	116
3.3	Sicherheitsprobleme	119
3.3.1	Sicherheitsprobleme von IP	119
3.3.2	Sicherheitsprobleme von ICMP	125
3.3.3	Sicherheitsprobleme von ARP	127
3.3.4	Sicherheitsprobleme von UDP und TCP	131
3.4	Sicherheitsprobleme von Netzdiensten	135
3.4.1	Domain Name Service (DNS)	136
3.4.2	Network File System (NFS)	145
3.4.3	Network Information System (NIS)	150
3.4.4	Weitere Dienste	152
3.5	Web-Anwendungen	157
3.5.1	World Wide Web (WWW)	157
3.5.2	Sicherheitsprobleme	163
3.5.3	OWASP Top-Ten Sicherheitsprobleme	172
3.6	Analysetools und Systemhärtung	181
4	**Security Engineering**	**189**
4.1	Entwicklungsprozess	190
4.1.1	Allgemeine Konstruktionsprinzipien	190
4.1.2	Phasen	191
4.1.3	BSI-Sicherheitsprozess	192
4.2	Strukturanalyse	196
4.3	Schutzbedarfsermittlung	198
4.3.1	Schadensszenarien	199
4.3.2	Schutzbedarf	201
4.4	Bedrohungsanalyse	203
4.4.1	Bedrohungsmatrix	203
4.4.2	Bedrohungsbaum	205
4.5	Risikoanalyse	210
4.5.1	Attributierung	211
4.5.2	Penetrationstests	216
4.6	Sicherheitsarchitektur und Betrieb	218
4.6.1	Sicherheitsstrategie und Sicherheitsmodell	218

4.6.2 Systemarchitektur und Validierung . . . 219
4.6.3 Aufrechterhaltung im laufenden Betrieb . . . 219
4.7 Sicherheitsgrundfunktionen . . . 220
4.8 Realisierung der Grundfunktionen . . . 224
4.9 Security Development Lifecycle (SDL) . . . 226
4.9.1 Die Entwicklungsphasen . . . 227
4.9.2 Bedrohungs- und Risikoanalyse . . . 228

5 Bewertungskriterien 233
5.1 TCSEC-Kriterien . . . 233
5.1.1 Sicherheitsstufen . . . 234
5.1.2 Kritik am Orange Book . . . 235
5.2 IT-Kriterien . . . 237
5.2.1 Mechanismen . . . 237
5.2.2 Funktionsklassen . . . 238
5.2.3 Qualität . . . 238
5.3 ITSEC-Kriterien . . . 239
5.3.1 Evaluationsstufen . . . 240
5.3.2 Qualität und Bewertung . . . 241
5.4 Common Criteria . . . 242
5.4.1 Überblick über die CC . . . 243
5.4.2 CC-Funktionsklassen . . . 247
5.4.3 Schutzprofile . . . 249
5.4.4 Vertrauenswürdigkeitsklassen . . . 252
5.5 Zertifizierung . . . 258

6 Sicherheitsmodelle 261
6.1 Modell-Klassifikation . . . 261
6.1.1 Objekte und Subjekte . . . 262
6.1.2 Zugriffsrechte . . . 263
6.1.3 Zugriffsbeschränkungen . . . 264
6.1.4 Sicherheitsstrategien . . . 264
6.2 Zugriffskontrollmodelle . . . 266
6.2.1 Zugriffsmatrix-Modell . . . 266
6.2.2 Rollenbasierte Modelle . . . 274
6.2.3 Chinese-Wall Modell . . . 282
6.2.4 Bell-LaPadula Modell . . . 287
6.3 Informationsflussmodelle . . . 294
6.3.1 Verbands-Modell . . . 294
6.4 Fazit und Ausblick . . . 298

7 Kryptografische Verfahren 301
7.1 Einführung . . . 301

7.2	Steganografie	303
7.2.1	Linguistische Steganografie	304
7.2.2	Technische Steganografie	305
7.3	Grundlagen kryptografischer Verfahren	307
7.3.1	Kryptografische Systeme	307
7.3.2	Anforderungen	312
7.4	Informationstheorie	314
7.4.1	Stochastische und kryptografische Kanäle	315
7.4.2	Entropie und Redundanz	317
7.4.3	Sicherheit kryptografischer Systeme	318
7.5	Symmetrische Verfahren	324
7.5.1	Permutation und Substitution	324
7.5.2	Block- und Stromchiffren	325
7.5.3	Betriebsmodi von Blockchiffren	330
7.5.4	Data Encryption Standard	336
7.5.5	AES	346
7.6	Asymmetrische Verfahren	352
7.6.1	Eigenschaften	352
7.6.2	Das RSA-Verfahren	356
7.7	Kryptoanalyse	368
7.7.1	Klassen kryptografischer Angriffe	368
7.7.2	Substitutionschiffren	370
7.7.3	Differentielle Kryptoanalyse	372
7.7.4	Lineare Kryptoanalyse	374
7.8	Kryptoregulierung	375
7.8.1	Hintergrund	375
7.8.2	Internationale Regelungen	376
7.8.3	Kryptopolitik in Deutschland	379
8	**Hashfunktionen und elektronische Signaturen**	**381**
8.1	Hashfunktionen	381
8.1.1	Grundlagen	382
8.1.2	Blockchiffren-basierte Hashfunktionen	388
8.1.3	Dedizierte Hashfunktionen	389
8.1.4	Message Authentication Code	393
8.2	Elektronische Signaturen	398
8.2.1	Anforderungen	398
8.2.2	Erstellung elektronischer Signaturen	400
8.2.3	Digitaler Signaturstandard (DSS)	404
8.2.4	Signaturgesetz	408
8.2.5	Fazit und Ausblick	414

9 Schlüsselmanagement 417
9.1 Zertifizierung 417
9.1.1 Zertifikate 418
9.1.2 Zertifizierungsstelle 419
9.1.3 Public-Key Infrastruktur 423
9.2 Schlüsselerzeugung und -aufbewahrung 431
9.2.1 Schlüsselerzeugung 431
9.2.2 Schlüsselspeicherung und -vernichtung 434
9.3 Schlüsselaustausch 437
9.3.1 Schlüsselhierarchie 437
9.3.2 Naives Austauschprotokoll 439
9.3.3 Protokoll mit symmetrischen Verfahren 441
9.3.4 Protokoll mit asymmetrischen Verfahren 445
9.3.5 Leitlinien für die Protokollentwicklung 447
9.3.6 Diffie-Hellman Verfahren 449
9.4 Schlüsselrückgewinnung 455
9.4.1 Systemmodell 456
9.4.2 Grenzen und Risiken 461

10 Authentifikation 465
10.1 Einführung 465
10.2 Authentifikation durch Wissen 468
10.2.1 Passwortverfahren 468
10.2.2 Authentifikation in Unix 482
10.2.3 Challenge-Response-Verfahren 488
10.2.4 Zero-Knowledge-Verfahren 492
10.3 Biometrie 495
10.3.1 Einführung 495
10.3.2 Biometrische Techniken 498
10.3.3 Biometrische Authentifikation 501
10.3.4 Fallbeispiel: Fingerabdruckerkennung 503
10.3.5 Sicherheit biometrischer Techniken 507
10.4 Authentifikation in verteilten Systemen 511
10.4.1 RADIUS 511
10.4.2 Remote Procedure Call 516
10.4.3 Secure RPC 518
10.4.4 Kerberos-Authentifikationssystem 521
10.4.5 Authentifikations-Logik 531

11 Digitale Identität 541
11.1 Smartcards 541
11.1.1 Smartcard-Architektur 542

11.1.2 Betriebssystem und Sicherheitsmechanismen 545
11.1.3 Fallbeispiele 549
11.1.4 Smartcard-Sicherheit 552
11.2 Elektronische Identifikationsausweise 556
11.2.1 Elektronischer Reisepass (ePass) 557
11.2.2 Elektronischer Personalausweis (nPA) 577
11.3 Trusted Computing 601
11.3.1 Trusted Computing Platform Alliance 602
11.3.2 TCG-Architektur 604
11.3.3 TPM 609
11.3.4 Sicheres Booten 623

12 Zugriffskontrolle 635
12.1 Einleitung 635
12.2 Speicherschutz 636
12.2.1 Betriebsmodi und Adressräume 637
12.2.2 Virtueller Speicher 638
12.3 Objektschutz 642
12.3.1 Zugriffskontrolllisten 643
12.3.2 Zugriffsausweise 647
12.4 Zugriffskontrolle in Unix 653
12.4.1 Identifikation 653
12.4.2 Rechtevergabe 654
12.4.3 Zugriffskontrolle 659
12.5 Zugriffskontrolle unter Windows 663
12.5.1 Architektur-Überblick 663
12.5.2 Sicherheitssubsystem 665
12.5.3 Datenstrukturen zur Zugriffskontrolle 668
12.5.4 Zugriffskontrolle 673
12.6 Verschlüsselnde Dateisysteme 676
12.6.1 Klassifikation 678
12.6.2 Encrypting File System (EFS) 680
12.7 Systembestimmte Zugriffskontrolle 686
12.8 Sprachbasierter Schutz 689
12.8.1 Programmiersprache 689
12.8.2 Übersetzer und Binder 692
12.9 Java-Sicherheit 698
12.9.1 Die Programmiersprache 699
12.9.2 Sicherheitsarchitektur 700
12.9.3 Java-Sicherheitsmodelle 705

13 Sicherheit in Netzen 713
13.1 Firewall-Technologie . . . 714
13.1.1 Einführung . . . 714
13.1.2 Paketfilter . . . 717
13.1.3 Proxy-Firewall . . . 731
13.1.4 Applikationsfilter . . . 735
13.1.5 Architekturen . . . 739
13.1.6 Risiken und Grenzen . . . 742
13.2 OSI-Sicherheitsarchitektur . . . 748
13.2.1 Sicherheitsdienste . . . 748
13.2.2 Sicherheitsmechanismen . . . 751
13.3 Sichere Kommunikation . . . 756
13.3.1 Verschlüsselungs-Layer . . . 758
13.3.2 Virtual Private Network (VPN) . . . 765
13.4 IPSec . . . 770
13.4.1 Überblick . . . 772
13.4.2 Security Association und Policy-Datenbank . . . 774
13.4.3 AH-Protokoll . . . 779
13.4.4 ESP-Protokoll . . . 782
13.4.5 Schlüsselaustauschprotokoll IKE . . . 786
13.4.6 Sicherheit von IPSec . . . 791
13.5 SSL/TLS . . . 796
13.5.1 Überblick . . . 798
13.5.2 Handshake-Protokoll . . . 801
13.5.3 Record-Protokoll . . . 804
13.5.4 Sicherheit von SSL/TLS . . . 807
13.6 Sichere Anwendungsdienste . . . 812
13.6.1 Elektronische Mail . . . 813
13.6.2 Elektronischer Zahlungsverkehr . . . 831
13.7 Service-orientierte Architektur . . . 839
13.7.1 Konzepte und Sicherheitsanforderungen . . . 839
13.7.2 Web-Services . . . 842
13.7.3 Web-Service Sicherheitsstandards . . . 847
13.7.4 SAML . . . 854
13.7.5 Offene Fragen . . . 859

14 Sichere mobile und drahtlose Kommunikation 863
14.1 Einleitung . . . 863
14.1.1 Heterogenität der Netze . . . 864
14.1.2 Entwicklungsphasen . . . 865
14.2 GSM . . . 868
14.2.1 Grundlagen . . . 868

14.2.2 GSM-Grobarchitektur . . . 869
14.2.3 Identifikation und Authentifikation . . . 870
14.2.4 Gesprächsverschlüsselung . . . 875
14.2.5 Sicherheitsprobleme . . . 877
14.2.6 GPRS . . . 881
14.3 UMTS . . . 883
14.3.1 UMTS-Sicherheitsarchitektur . . . 884
14.3.2 Authentifikation und Schlüsselvereinbarung . . . 887
14.3.3 Vertraulichkeit und Integrität . . . 890
14.4 Funk-LAN (WLAN) . . . 892
14.4.1 Grundlagen . . . 893
14.4.2 WLAN-Sicherheitsprobleme . . . 900
14.4.3 WEP . . . 905
14.4.4 WPA und 802.11i . . . 919
14.5 Bluetooth . . . 934
14.5.1 Einordnung und Abgrenzung . . . 935
14.5.2 Technische Grundlagen . . . 938
14.5.3 Sicherheitsarchitektur . . . 943
14.5.4 Schlüsselmanagement . . . 948
14.5.5 Authentifikation . . . 953
14.5.6 Bluetooth-Sicherheitsprobleme . . . 956
14.5.7 Secure Simple Pairing . . . 959
14.6 Long Term Evolution (LTE) und SAE . . . 963
14.6.1 EPC und LTE . . . 965
14.6.2 Interworking . . . 967
14.6.3 Sicherheitsarchitektur und Sicherheitsdienste . . . 968
14.6.4 Sicheres Interworking . . . 975

Literaturverzeichnis **979**

Abkürzungsverzeichnis **995**

Index **1005**

1 Einführung

IKT

Informations- und Kommunikationstechnologie (IKT) ist heute in nahezu allen Bereichen von zentraler Bedeutung. Eingebettete Systeme, Machine-to-Machine (M2M) Kommunikation, Vernetzung, aber auch on-demand beziehbare Mehrwertdienste aus der Cloud sind zentrale Wachstumstreiber einer immer stärker digitalisierten Wirtschaft. Der IT-Sicherheit kommt hierbei eine Schlüsselrolle zu.

IT-Sicherheit

IT-Sicherheit hat die Aufgabe, Unternehmen und deren Werte (Know-How, Kundendaten, Personaldaten) zu schützen und wirtschaftliche Schäden, die durch Vertraulichkeitsverletzungen, Manipulationen oder auch Störungen der Verfügbarkeit von Diensten des Unternehmens entstehen können, zu verhindern. Da eine vollständige Vermeidung oder Verhinderung von Angriffen in der Praxis nicht möglich ist, umfasst das Gebiet der IT-Sicherheit insbesondere auch Maßnahmen und Konzepte, um das Ausmaß potentieller Schäden, die durch Sicherheitsvorfälle entstehen können, zu reduzieren und damit die Risiken beim Einsatz von IKT-Systemen zu verringern. Schwachstellen und konkrete Angriffe oder Angriffsversuche auf IKT-Systeme müssen frühzeitig und mit möglichst hoher Präzision erkannt werden können und auf eingetretene Schadensfälle muss mit geeigneten technischen Maßnahmen reagiert werden. Techniken der Angriffserkennung und -Reaktion gehören deshalb ebenso zur IT-Sicherheit, wie methodische Grundlagen, um IKT-Systeme so zu entwickeln, dass sie qua Design ein hohes Maß an Sicherheit bieten. Man spricht in diesem Zusammenhang auch oft von *Secure by Design*. Gleichzeitig dient IT-Sicherheit als Enabling-Technologie. Sie ermöglicht die Entwicklung neuer, vertrauenswürdiger Anwendungen und Dienstleistungen verbunden mit innovativen Geschäftsmodellen beispielsweise im Gesundheitsbereich, bei Automotive-Anwendungen oder aber auch in zukünftigen, intelligenten Umgebungen, wie Smart Grid, Smart Factory, Smart Health oder auch Smart Cities.

IT-Sicherheit für Sicherheit

Zudem dienen die Technologien und Verfahren der IT-Sicherheit dazu, die Sicherheit in einem allgemeineren Sinn (u.a. im Sinne öffentlicher Sicherheit) zu erhöhen. Zur Überwachung von beispielsweise kritischen Infrastrukturen, wie Kraftwerken, Produktionsanlagen, Transportleitsysteme etc. wird bereits heute eine Vielzahl von Sensoren eingesetzt, die

kontinuierlich Umgebungsdaten erfassen, Daten austauschen und die das Ablaufverhalten bzw. den operativen Betrieb komplexer Anlagen permanent überwachen. Probleme und auffälliges Verhalten werden an zumeist zentrale Leitstellen gemeldet, so dass auf der Basis dieser Steuerungsdaten kontrollierend in kritische Abläufe (zum Teil vollständig automatisiert) eingegriffen wird. Es ist unmittelbar klar, dass die zur Steuerung und Kontrolle verwendeten Daten vor Manipulationen zu schützen sind. Sie müssen zudem rechtzeitig und vollständig vorliegen. Vielfach ist es zudem aus Gründen des Datenschutzes auch notwendig, die Daten vertraulich zu verarbeiten, man denke nur an die aktuelle Diskussion zu den digital erfassten und per Datenkommunikation übermittelten Stromverbrauchsdaten privater Haushalte. IT-Sicherheitskonzepte sind die wichtige Basis, damit die zur Überwachung und Steuerung der Anlagen eingesetzten IKT-Komponenten auch verlässlich für sicherheitskritische Anwendungen eingesetzt werden können.

Das vorliegende Buch hat zum Ziel, fundierte Kenntnisse über Funktionsweise und Wirksamkeit existierender Maßnahmen zur Absicherung von IT Systemen zu vermitteln. Das vorliegende Kapitel führt hierzu die notwendigen generellen Grundlagen und Begrifflichkeiten ein.

Kapitelüberblick

In Abschnitt 1.1 werden die grundlegenden Begriffe und Definitionen eingeführt, die für den Bereich sicherer IT-Systeme von Bedeutung sind. Datenobjekte und Informationen gehören zu den zu schützenden Gütern eines technischen Systems und bilden die Grundlage zur Festlegung der Schutzziele. Das sind Sicherheitsanforderungen, die an ein System gestellt werden und die durch die Sicherheitseigenschaften des Systems schlussendlich zu gewährleisten sind. Die wichtigsten dieser Eigenschaften werden in Abschnitt 1.2 vorgestellt. Aktionen und Ereignisse, die die Sicherheitseigenschaften eines Systems in Frage stellen, sind Angriffe auf das System. Angriffe nutzen Schwachstellen des Systems aus. Abschnitt 1.3 erklärt die in diesem Kontext interessierenden Begriffe und erläutert deren Zusammenhänge. Mit der Zunahme der Computerkriminalität gewinnen Maßnahmen an Bedeutung, durch die elektronische Spuren eines digitalen Einbruchs gesichert werden können, um sie in einem Rechtsstreit oder zur Strafverfolgung verwerten zu können. Mit diesem relativ jungen Zweig der IT-Sicherheit, der Computer Forensik, beschäftigt sich Abschnitt 1.4. Die einzuhaltenden Sicherheitseigenschaften eines IT-Systems werden formal oder informell in einer Sicherheitsstrategie festgelegt. Abschnitt 1.5 gibt einen ersten Einblick in Sicherheitsstrategien. Abschnitt 1.6 fasst abschließend die wichtigsten Komponenten eines sicheren IT-Systems zusammen.
Einen guten Einstieg in die Problematik der IT-Sicherheit liefern die Bücher von B. Schneier [188] und Charles Pfleeger [164].

1.1 Grundlegende Begriffe

Dieser Abschnitt dient dazu, die für das Folgende wesentlichen Begriffe und Termini einzuführen. Da in der Literatur keine einheitliche Begriffsbildung verwendet wird, ist eine entsprechende Festlegung und Einführung notwendig. In diesem Buch beschäftigen wir uns mit der Sicherheit informationstechnischer Systeme, kurz IT-Systeme, so dass wir zunächst den Begriff des IT-Systems präzisieren.

Definition 1.1 (IT-System)

IT-System

Ein IT-System ist ein geschlossenes oder offenes, dynamisches technisches System mit der Fähigkeit zur Speicherung und Verarbeitung von Informationen.

□

geschlossen

Unter einem geschlossenen System verstehen wir ein System, das auf der Technologie eines Herstellers aufbaut, zu Konkurrenzprodukten nicht kompatibel ist und dessen Ausdehnung sich auf einen bestimmten Teilnehmerkreis und ein bestimmtes räumliches Gebiet beschränkt. Geschlossene Systeme sind in der Regel homogen und werden zentral verwaltet. Unter offenen Systemen verstehen wir vernetzte, physisch verteilte Systeme, die sich an Standards zum Informationsaustausch mit anderen Systemen orientieren. Jedes Teilsystem ist offen für die Kommunikation mit anderen Systemen. Offene Systeme sind meist durch heterogene Hardwarekomponenten und Betriebssysteme gekennzeichnet, für die keine zentrale Administration und Verwaltung zur Verfügung steht.

offen

Die im vorliegenden Buch behandelten Sicherheitseigenschaften von IT-Systemen gelten sowohl für geschlossene als auch für offene Systeme. Die Offenheit eines Systems erfordert jedoch erweiterte und andere Realisierungsmaßnahmen zur Gewährleistung der Sicherheitseigenschaften als sie in einer geschlossenen Umgebung benötigt werden. Der von uns verwendete Systembegriff bezieht sich auf den weiten Bereich der Systeme der Informations- und Kommunikationstechnik (IuK), wozu Betriebssysteme und Kommunikationssysteme ebenso zählen wie Datenverarbeitungssysteme für spezifische Anwendungsbereiche (u.a. ein Krankenhausinformationssystem, eine Homebanking-Anwendung oder ein Telearbeitsplatz).

soziotechnisches System

IT-Systeme sind Bestandteile soziotechnischer Systeme. Das heißt, sie sind eingebettet in gesellschaftliche, unternehmerische und auch politische Strukturen und werden von Benutzern mit sehr unterschiedlichem technischen Know-how und für sehr unterschiedliche Zwecke genutzt. Bei der Betrachtung der technischen Aspekte der IT-Sicherheit muss man sich die-

ser Einbettung bewusst sein. Fragen der Nutzbarkeit und Akzeptanz von Sicherheitstechnologien sind hier ebenso zu betrachten, wie die Einhaltung von gesetzlichen und organisatorischen Regelungen und Vorschriften (u.a. Bundesdatenschutzgesetz, unternehmensspezifische Geheimhaltungsvorschriften, Betriebsverfassungsgesetz). Das vorliegende Buch beschränkt sich jedoch in erster Linie auf die Behandlung der technischen Aspekte der IT-Sicherheit.

Information und Datenobjekte

Objekt

IT-Systeme speichern und verarbeiten Informationen. Die Information ist aber ein Abstraktum, das in Form von Daten bzw. Datenobjekten repräsentiert wird. Wir unterscheiden zwischen passiven Objekten (z.B. Datei, Datenbankeintrag) mit der Fähigkeit, Informationen zu speichern, und aktiven Objekten (z.B. Prozesse) mit der Fähigkeit, sowohl Informationen zu speichern als auch zu verarbeiten. Informationen und die Objekte, die sie repräsentieren, sind schützenswerte Güter (engl. *asset*) eines Systems.

Information

Die Information, die durch ein Datum repräsentiert wird, ergibt sich aus einer festgelegten Interpretationsvorschrift. So kann beispielsweise ein Integer-Datenobjekt einen numerischen Wert besitzen, der, abhängig von der Interpretationsvorschrift, zum Beispiel den Kontostand eines Bankkontos, eine Speicheradresse oder aber auch eine Entfernungsangabe repräsentiert. Ferner ist es möglich, dass ein und dieselbe Information in unterschiedlichen Datenobjekten repräsentiert ist. Ein geheimes Passwort kann zum Beispiel als eine Folge von Binärzeichen auf der Festplatte in einem Plattenblock (Datenobjekt) gespeichert sein, oder nach der Eingabe durch den Benutzer im Tastatureingabepuffer (Datenobjekt) stehen oder als Nutzdaten in einem IP-Nachrichtpaket (Datenobjekt) enthalten sein. Will man also die Information als abstraktes Gut schützen, muss man sich darüber klar werden, in welchen verschiedenen Datenobjekten diese Information im System repräsentiert wird. Eine Nichtbeachtung dieser Zusammenhänge hat in der Vergangenheit wiederholt zu Sicherheitsproblemen geführt. Ein Beispiel dafür ist das verschlüsselnde Dateisystem EFS unter Windows 2000, Vista oder auch Windows 7 (siehe auch Abschnitt 12.6.2). EFS stellt Funktionen zur Verfügung, über die sensible Inhalte von Dateien nach der Bearbeitung verschlüsselt auf der Festplatte abgelegt werden können, damit sie nicht mehr im Klartext im System verfügbar sind. In den ersten Versionen dieser Software wurde jedoch vernachlässigt, dass das Speichermanagement des Betriebssystems aus Effizienzgründen häufig genutzte Daten in einem Schnellzugriffspeicher (engl. *cache*) ablegt. Die sensible Information war somit auch in Cache-Blöcken repräsentiert und aufgrund fehlender Bereinigungsoperationen auch nach dem Schließen der Datei noch für einige Zeit im Klartext im System vorhanden.

Datenobjekte sind über wohl definierte Schnittstellen in Form von Operationen bzw. Methoden von anderen Objekten des Systems oder von der Umwelt des Systems benutzbar. Zur Umwelt gehören insbesondere seine Benutzer. Die Benutzer eines Systems und alle Objekte, die im Auftrag von Benutzern im System aktiv sein können, wie z.B. Prozesse, Server und Prozeduren, werden als die Subjekte des Systems bezeichnet.

Subjekt

Eine Interaktion zwischen einem Subjekt und einem Objekt in einem System, durch die ein Informationsfluss zwischen Subjekt und Objekt auftritt, nennen wir einen Zugriff auf die Information. Jeder Zugriff auf ein Datenobjekt ist gleichzeitig auch ein Zugriff auf die dadurch repräsentierte Information. Für den Zugriff auf die zu schützende Information bzw. auf die sie repräsentierenden Daten eines IT-Systems sind Zugriffsrechte festzulegen und an die Subjekte zu vergeben. Besitzt ein Subjekt die Berechtigung zum Zugriff auf eine Information bzw. auf ein Datenobjekt, so sagen wir, dass das Subjekt zu diesem Zugriff autorisiert ist.

Zugriffsrecht

Autorisierung

Informationskanäle

Die in einem System potentiell auftretenden Kanäle, über die Informationen fließen können, werden nach Speicherkanälen, legitimen und verdeckten Kanälen klassifiziert. Die legitimen Kanäle sind diejenigen, die ein Subjekt in der Regel für den Informationsaustausch nutzt (u.a. Nachrichten und Parameter bei Operationsaufrufen). Unter Speicherkanälen versteht man Objekte, die von Subjekten gemeinsam benutzt werden können (u.a. Dateien). Verdeckte Kanäle (engl. *covert channel*) sind solche, die nicht für einen Informationstransfer vorgesehen sind, aber dazu missbraucht werden können. Zum Beispiel kann durch die Veränderung der Ausführungszeit eines Programms eine verdeckte Information an den Beobachter der Ausführungszeiten übermittelt werden. So könnte beispielsweise die Erhöhung der Ausführungszeit signalisieren, dass ein Konkurrent ein preiswerteres Angebot erstellt hat als der Empfänger dieses Signals. Aufgrund der Komplexität heutiger Software-Systeme mit ihrer Vielzahl von direkten und indirekten Wechselwirkungen zwischen verschiedenen Systemkomponenten sind verdeckte Kanäle nicht vollständig zu verhindern. Durch eine methodische Systemkonstruktion kann jedoch dafür gesorgt werden, dass die Bandbreite verdeckter Kanäle, also die Menge der Informationen, die über diese Kanäle transferiert werden kann, beschränkt ist. So ist der Informationsgehalt eines 1-Bit Signals (Ja/Nein) selbstverständlich deutlich geringer, als der einer Nachricht von mehreren Byte Länge oder von einer Datei mit einer Größe von mehreren KByte.

legitimer Kanal

verdeckter Kanal

Auf der Basis der getroffenen Festlegungen können wir nun die wichtigsten Sicherheitsbegriffe einführen.

Definition 1.2 (Sicherheit)

funktionssicher Unter Funktionssicherheit (engl. *safety*) eines Systems verstehen wir die Eigenschaft, dass die realisierte Ist-Funktionalität der Komponenten mit der spezifizierten Soll-Funktionalität übereinstimmt. Ein funktionssicheres System nimmt keine funktional unzulässigen Zustände an. Anders formuliert verstehen wir unter der Funktionssicherheit eines Systems, dass es unter allen (normalen) Betriebsbedingungen funktioniert.

informationssicher Die Informationssicherheit (engl. *security*) ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keiner unautorisierten Informationsveränderung oder -gewinnung führen.

Datensicherheit Die Datensicherheit (engl. *protection*) ist die Eigenschaft eines funktionssicheren Systems, nur solche Systemzustände anzunehmen, die zu keinem unautorisierten Zugriff auf Systemressourcen und insbesondere auf Daten führen. Damit umfasst die so beschriebene Sicherheit der Daten insbesondere auch Maßnahmen zur Datensicherung (engl. *backup*), also den Schutz vor Datenverlusten durch Erstellung von Sicherungskopien.

Datenschutz Unter dem Begriff Datenschutz im engeren Sinn (engl. *privacy*), wie er unter anderem durch das deutsche Bundesdatenschutzgesetz (BDSG) [52] festgelegt wird, versteht man die Fähigkeit einer natürlichen Person, die Weitergabe von Informationen, die sie persönlich betreffen, zu kontrollieren. In diesen Bereich fallen insbesondere Sicherheitsanforderungen, die der deutsche Gesetzgeber durch das informationelle Selbstbestimmungsrecht geregelt hat.

□

Mit den getroffenen Festlegungen können wir zwischen Systemen, die vordringlich auf den Schutz der verarbeiteten Informationen (Informationssicherheit) und solchen, die auf den Schutz der Daten als deren Repräsentanten (Datensicherheit) abzielen, unterscheiden. Damit ist eine Präzisierung vorgenommen, die in den meisten Abhandlungen zu dieser Thematik fehlt. Herkömmlicherweise wird unter der Sicherheit eines Systems stets dessen Datensicherheit verstanden. Die eingeführte Definition ist auch eine Präzisierung der Sicherheits-Begriffe im Vergleich zu dem allgemeinen, durch die ISO (International Standards Organization) geprägten Sicherheitsbegriff [104]. Dort wird die Sicherheit (*security*) als „die Minimierung der Verwundbarkeit von Werten und Ressourcen“ definiert.

Aus den Festlegungen von Definition 1.2 ist ferner die wesentliche Erkenntnis abzuleiten, dass die Sicherstellung der Informations- und Datensicherheit eines IT-Systems ein Prozess ist (vgl. auch Kapitel 4.1), der einer ständigen Überprüfung und Revision unterliegt, da sich Systemeigenschaften über die Zeit dynamisch ändern können. Die Definition 1.2 verdeutlicht zudem, dass

die Funktionssicherheit die Grundlage für die Informations- bzw. Datensicherheit eines Systems ist. Die Funktionssicherheit ist eng verwandt mit den Begriffen der Zuverlässigkeit bzw. der Verlässlichkeit.

Definition 1.3 (Verlässlichkeit)

Verlässlichkeit

Unter der Verlässlichkeit (engl. *dependability*) eines Systems verstehen wir die Eigenschaft, keine unzulässigen Zustände anzunehmen (Funktionssicherheit) und zu gewährleisten, dass die spezifizierte Funktion zuverlässig (engl. *reliability*) erbracht wird.

□

Safety

Zur Gewährleistung von Funktionssicherheit (Safety) setzt man Konzepte und Verfahren ein, die darauf abzielen, die Verlässlichkeit von IT-Systemen zu gewährleisten. Es handelt sich dabei im Wesentlichen um Maßnahmen zur Abwehr von solchen Bedrohungen, die durch das technische Fehlverhalten des IT-Systems selber (von innen) entstehen. Derartige Bedrohungen ergeben sich insbesondere durch Programmierfehler, die mit Techniken der Programmvalidierung oder -verifikation aufzudecken sind.

Im vorliegenden Buch beschäftigen wir uns mit der Sicherheit technischer Systeme im Sinne der englischen Begriffe *Security* und *Protection*. Behandelt werden Konzepte und Maßnahmen zur Abwehr von Bedrohungen, die durch unberechtigte Zugriffe auf die zu schützenden Güter des IT-Systems entstehen und im Wesentlichen von außen erfolgen. Anzumerken ist, dass die Grenzen zwischen Safety- und Security-Fragestellungen fließend sind und es durchaus Überlappungen gibt.

1.2 Schutzziele

Informationen bzw. Daten sind zu schützende Güter informationssicherer bzw. datensicherer Systeme. Der Zugriff auf diese ist zu beschränken und zu kontrollieren, so dass nur dazu autorisierten Subjekten ein Zugriff gewährt wird. Die Schutzziele, die diese Anforderungen präzisieren, sind die Datenintegrität und Informationsvertraulichkeit. Zugreifende Subjekte müssen eindeutig identifiziert und ihre Identität muss verifiziert sein. Die entsprechende Eigenschaft nennt man die Authentizität von Subjekten. Ist ein Subjekt authentifiziert und berechtigt, also autorisiert, einen Zugriff auf ein Objekt bzw. eine Information durchzuführen, dann sollte das System gewährleisten, dass dieser Zugriff auch möglich ist; man spricht von der Eigenschaft der Verfügbarkeit. Hat ein Subjekt einen Zugriff bzw. eine Aktion durchgeführt, so ist es vielfach notwendig, dass auch noch im Nachhinein die Urheberschaft des Zugriffs bzw. der Aktion eindeutig dem entsprechen-

den Subjekt zuordenbar ist. Man spricht hier von der Verbindlichkeit oder Zuordenbarkeit des Systems. Die angesprochenen Schutzziele werden nun im Folgenden präzisiert.

Definition 1.4 (Authentizität)

Authentizität

Unter der Authentizität eines Objekts bzw. Subjekts (engl. *authenticity*) verstehen wir die Echtheit und Glaubwürdigkeit des Objekts bzw. Subjekts, die anhand einer eindeutigen Identität und charakteristischen Eigenschaften überprüfbar ist.

□

Authentifikation

Die Authentizität eines Subjekts bzw. Objekts wird durch Maßnahmen zur Authentifikation (engl. *authentication*) überprüft. Dazu muss nachgewiesen werden, dass eine behauptete Identität eines Objekts oder Subjekts mit dessen charakterisierenden Eigenschaften übereinstimmt.

Subjekt-Auth.

In herkömmlichen Systemen wird eine Authentifikation meist nur für Benutzer als Subjekte durchgeführt. Die Identifikation basiert auf der Vergabe von eindeutigen Benutzerkennungen (engl. *account*) oder Benutzernamen. Charakterisierende Eigenschaften zum Nachweis der Identität sind beispielsweise Passwörter, deren Kenntnis der Benutzer beim Systemzugang nachweisen muss, oder biometrische Merkmale wie Fingerabdrücke. Identitätsnachweise werden häufig allgemein als Credentials bezeichnet, womit man von dem spezifischen, zum Identitätsnachweis tatsächlich verwendeten Verfahren abstrahiert.

Objekt-Authentizität

Mit dem Übergang zu offenen Systemen werden auch zunehmend Maßnahmen erforderlich, die die Authentizität von Objekten, wie beispielsweise Web-Server, Access Points (bei 802.11 WLAN) oder Code nachweisen. Hierbei beschränkt man sich jedoch i.d.R. auf einfache Mechanismen, nämlich kryptografische Verfahren (vgl. Kapitel 7), um die Echtheit von Daten zu überprüfen, die über ein unsicheres Transportmedium wie dem Internet übertragen werden. Diese Echtheitsprüfung beschränkt sich auf einen Ursprungs- bzw. Urhebernachweis, ohne Aussagen über die Funktionalität des Objekts zu treffen. Die Authentizität eines Objekts im engeren Sinn würde den Nachweis erfordern, dass seine spezifizierte Funktionalität mit seiner tatsächlich erbrachten übereinstimmt. Entsprechende Nachweise, die z.B. unter Verwendung der Proof-Carrying-Code Technik [150] erstellt werden, sind sehr schwierig zu führen und werden in der Praxis noch nicht eingesetzt. Hier sind noch weitere, verstärkte Forschungsaktivitäten notwendig, da gerade mit der ansteigenden Mobilität von Daten und von mobilem Code und Apps (vgl. Kapitel 2.7) wie zum Beispiel der Java-Applets oder Apps für Android und Apple iPhone, ein Authentizitätsnachweis, der neben

einem Ursprungsnachweis auch eine Aussage über eine korrekte Funktionalität beinhaltet, für die Sicherheit offener Systeme dringend erforderlich ist.

Definition 1.5 (Datenintegrität)

Integrität

Wir sagen, dass das System die Datenintegrität (engl. *integrity*) gewährleistet, wenn es Subjekten nicht möglich ist, die zu schützenden Daten unautorisiert und unbemerkt zu manipulieren.

□

Rechtefestlegung

Die Eigenschaft der Datenintegrität erfordert zum einen die Festlegung von Rechten zur Nutzung von Daten. Beispiele hierfür sind Lese- oder Schreibberechtigungen für Dateien oder das Recht, von einem bestimmten Konto einen Betrag bis zu einer festgelegten Obergrenze abheben zu dürfen. Zum anderen sind Rechte an Subjekte zu vergeben, so dass diese autorisiert sind, die entsprechenden Zugriffsrechte wahrzunehmen. Abhängig von den damit getroffenen Festlegungen können Integritätsaussagen unterschiedlicher Qualität gemacht werden. So wird beispielsweise durch die Vergabe von Schreibberechtigungen an Dateien die Möglichkeit zur Modifikation des Datei-Objekts nicht weiter beschränkt, so dass Subjekte zu beliebigen Manipulationen berechtigt sind. Auf dieser Basis sind nur eingeschränkte Aussagen über die Integrität im Sinne einer authentischen, korrekten Funktionalität des Daten-Objekts möglich. Legt man demgegenüber die Berechtigungen zur Nutzung von Objekten durch wohl definierte Methoden des Objekts fest, so werden die Nutzungsmöglichkeiten und damit die Manipulationsmöglichkeiten auf die Funktionalität dieser Zugriffsoperationen eingeschränkt. Auf formale Techniken zur Festlegung und Vergabe von Zugriffsrechten wird in Kapitel 6 eingegangen. Die benötigten Mechanismen und Verfahren zur Gewährleistung des Schutzzieles der Datenintegrität gehören zum Bereich der Zugriffskontrolle.

Manipulations-Erkennung

Definition 1.5 fordert, dass unautorisierte Manipulationen nicht unbemerkt bleiben dürfen. Das bedeutet, dass in Umgebungen, in denen eine solche Manipulation nicht a priori verhindert werden kann (z.B. in Netzen), Techniken erforderlich sind, mit deren Hilfe unautorisierte Manipulationen a posteriori erkennbar sind. Auf diese Weise kann verhindert werden, dass unautorisiert manipulierte Daten weiterverarbeitet werden und der mögliche Schaden begrenzt wird. Zur Erkennung von durchgeführten Datenveränderungen werden kryptografisch sichere Hashfunktionen (vgl. Kapitel 8.1) eingesetzt.

Definition 1.6 (Informationsvertraulichkeit)

Vertraulichkeit

Wir sagen, dass das System die Informationsvertraulichkeit (engl. *confidentiality*) gewährleistet, wenn es keine unautorisierte Informationsgewinnung ermöglicht.

□

Die Gewährleistung der Eigenschaft der Informationsvertraulichkeit erfordert in datensicheren Systemen die Festlegung von Berechtigungen und Kontrollen der Art, dass sichergestellt ist, dass Subjekte nicht unautorisiert Kenntnis von Informationen erlangen. In informationssicheren Systemen sind Maßnahmen zur Festlegung sowie zur Kontrolle zulässiger Informationsflüsse zwischen den Subjekten des Systems erforderlich, so dass ausgeschlossen werden kann, dass Information zu unautorisierten Subjekten „durchsickert". Das entsprechende Problem wird Confinement-Problem [125] genannt. Mit den Festlegungen zur Kontrolle der Informationsflüsse ist spezifiziert, welche Subjekte Kenntnis von welchen Informationen erlangen dürfen. Formale Techniken zur Formulierung zulässiger bzw. verbotener Informationskanäle werden mit den Informationsflussmodellen in Kapitel 6 eingeführt.

Confinement

Im Zusammenhang mit Datenbanksystemen ergeben sich spezifische Vertraulichkeitsprobleme, auf die wir im Rahmen dieses Buches jedoch nicht weiter eingehen werden. Die entsprechenden Probleme sind unter dem Stichwort Interferenz-Kontrolle (engl. *interference control*) bekannt (u.a. [58]). Hierbei geht es darum, dass aus der Kenntnis von Einzelinformationen weitere Informationen abgeleitet werden können, wobei diese abgeleitete Information jedoch dem betreffenden Subjekt eigentlich nicht zugänglich sein darf. Man denke hier beispielsweise an Datenbankanfragen, die Auskünfte über Einzelpersonen ermöglichen, wobei aber die Anonymität der Personen gewahrt sein muss. Falls es durch geschicktes Formulieren entsprechender Anfragen gelingt, sehr kleine Antwortmengen zu erhalten, sind Rückschlüsse auf spezifische Personen unter Umständen möglich.

Interferenz

Anforderungen an die Informationsvertraulichkeit im weiteren Sinn werden durch Verschlüsselungstechniken (vgl. Kapitel 7) erfüllt. Hierbei besteht das Ziel darin, die Daten geeignet zu transformieren, so dass unautorisierte Dritte nicht in der Lage sind, ohne den korrekten Entschlüsselungsschlüssel die Daten sinnvoll zu interpretieren.

Beispiel 1.1 (Unzulässiger Informationsfluss)

Informationsfluss

Die unterschiedlichen Anforderungen für datensichere und informationssichere Systeme ergeben sich aus dem beschriebenen Zusammenhang zwischen Information als Abstraktum und den Daten als deren Repräsentation.

Mit der Kontrolle der Datenzugriffe werden nämlich nicht notwendigerweise auch alle Informationszugriffe im erforderlichen Umfang kontrolliert. Man betrachte dazu beispielsweise ein Szenario mit zwei Benutzern `Bill` und `Joe`, zwei Datei-Objekten `Datei_1` und `Datei_2` und der Vertraulichkeitsanforderung, dass das Subjekt `Joe` keine Kenntnis über die durch die `Datei_1` repräsentierten Informationen erlangen darf.

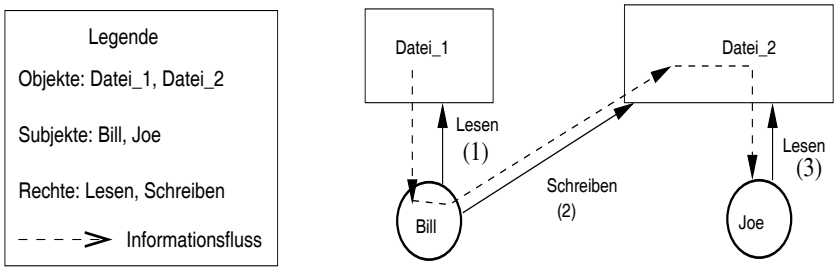

Abbildung 1.1: Unzulässiger Informationsfluss

Der Benutzer `Bill` sei berechtigt, schreibend auf `Datei_2` sowie lesend auf `Datei_1` zuzugreifen, während der Benutzer `Joe` nur lesend auf `Datei_2` zugreifen darf. Erfolgen nur autorisierte Zugriffe gemäß dieser Rechtevergabe, so kann dennoch ein unzulässiger Informationsfluss auftreten. Abbildung 1.1 skizziert die Situation, die eintritt, wenn der Benutzer `Bill` zunächst Informationen aus `Datei_1` ausliest (1) und diese anschließend in `Datei_2` schreibt (2). Daraufhin erhält Benutzer `Joe` auf diese Informationen lesenden (3) Zugriff.
▲

Zur Gewährleistung von Vertraulichkeitsanforderungen sind somit spezielle Maßnahmen erforderlich, die über eine reine Kontrolle der Zugriffe auf die Datenobjekte hinausgehen. Verwendet werden hierbei, neben kryptografischen Verfahren, insbesondere auch Labeling-Techniken, wodurch Datenobjekte eine spezielle Sicherheitseinstufung erhalten. Durch spezielle Kontrollen kann sichergestellt werden, dass Information, die z.B. als sensitiv eingestuft ist, nicht in unautorisierte Hände gelangt.

Für autorisierte und authentifizierte Subjekte ist zu gewährleisten, dass sie die ihnen zugebilligten Zugriffsrechte auch wahrnehmen können, d.h. dass das System die entsprechenden Leistungen (z.B. Zugriff auf einen Service, für dessen Nutzung der Kunde bezahlt hat) zur Verfügung stellt.

Definition 1.7 (Verfügbarkeit)

Verfügbarkeit

Wir sagen, dass das System die Verfügbarkeit (engl. *availability*) gewährleistet, wenn authentifizierte und autorisierte Subjekte in der Wahrnehmung ihrer Berechtigungen nicht unautorisiert beeinträchtigt werden können.

□

Da in einem IT-System in der Regel zu einem Zeitpunkt unterschiedliche Prozesse eines Benutzers oder auch verschiedener Benutzer um gemeinsame Ressourcen wie CPU-Zeit oder Ein- und Ausgabegeräte konkurrieren, kann es für die einzelnen Prozesse zu Ausführungsverzögerungen kommen. Verzögerungen, die aus „normalen“ Verwaltungsmaßnahmen resultieren (u.a. Prozess-Scheduling), werden als autorisierte Beeinträchtigungen betrachtet, die a priori noch keine Verletzung der Verfügbarkeit darstellen. Es ist klar, dass hier im Gegensatz zu den zuvor eingeführten Eigenschaften die Trennlinie zwischen unautorisierten und autorisierten Aktionen unscharf ist, da u.U. ein Angreifer durch autorisierte Zugriffe die Verwaltungsmaßnahmen des Systems beeinflussen und damit absichtlich einen Angriff auf die Verfügbarkeit hervorrufen kann. Dies kann beispielsweise durch das gezielte Monopolisieren der CPU mit einem hoch prioren Prozess geschehen. Entsprechend schwierig ist deshalb auch das Erkennen einer unautorisierten Beeinträchtigung. So kann ein hohes Datenaufkommen, das zu Stausituationen in einem Netz führt, sowohl auf normalen Datenverkehr als auch auf einen Angriff durch synthetisch erzeugte Nachrichten zurückzuführen sein.

Maßnahmen

Zur Gewährleistung der Verfügbarkeit sind Maßnahmen, wie die Einführung von Quoten, von Interesse, durch die die Nutzung von Systemressourcen wie beispielsweise CPU-Zeit oder Speicher reglementiert wird. Auf Maßnahmen zur Erfüllung von Verfügbarkeitsanforderungen gehen wir in diesem Buch jedoch nicht näher ein.

Schließlich bleibt noch die Eigenschaft der Verbindlichkeit zu präzisieren.

Definition 1.8 (Verbindlichkeit)

Verbindlichkeit

Wir sagen, dass das System die Verbindlichkeit bzw. Zuordenbarkeit (engl. *non repudiation*) einer Menge von Aktionen gewährleistet, wenn es nicht möglich ist, dass ein Subjekt im Nachhinein die Durchführung einer solchen Aktion abstreiten kann.

□

Verbindlichkeitseigenschaften sind besonders in dem rasant wachsenden Bereich des elektronischen Handels (engl. *Electronic Commerce*) und der elektronischen Geschäfte (engl. *Electronic Business*) von großer Bedeutung, um die Rechtsverbindlichkeit durchgeführter geschäftlicher Transaktionen

(Käufe, Verträge etc.) zu garantieren. Diese Anforderungen lassen sich durch den Einsatz digitaler Signaturen (vgl. Kapitel 8.2) erfüllen. Die Verbindlichkeit ist aber auch allgemein bei der Nutzung von Systemressourcen in Mehrbenutzersystemen oder auch zunehmend bei Grid- und Peer-to-Peer Computing von Interesse. Beispiele relevanter Aktionen sind hier der Verbrauch von Rechenzeit oder die Nutzung teurer Ein/Ausgabe-Geräte. Mit der Verbindlichkeitseigenschaft ist die Forderung nach Abrechenbarkeit (engl. *accountability*) unmittelbar verbunden. Dies erfordert Maßnahmen zur Überwachung (engl. *audit*) sowie zur Protokollierung einzelner Benutzeraktivitäten.

Accountability

Audit

Zusätzlich zu den betrachteten Sicherheitseigenschaften gewinnt im Kontext vernetzter und insbesondere mobiler Systeme zunehmend die Forderung, die Anonymität von Subjekten zu gewährleisten und deren Privatsphäre zu schützen, an Bedeutung.

Anonymität

Definition 1.9 (Anonymisierung und Pseudomisierung)

Unter der Anonymisierung versteht man das Verändern personenbezogener Daten der Art, dass die Einzelangaben über persönliche oder sachliche Verhältnisse nicht mehr oder nur mit einem unverhältnismäßig großen Aufwand an Zeit, Kosten und Arbeitskraft einer bestimmten oder bestimmbaren natürlichen Person zugeordnet werden können.

Anonymisierung

Eine schwächere Form der Anonymisierung stellt die Pseudomisierung dar. Dabei handelt es sich um das Verändern personenbezogener Daten durch eine Zuordnungsvorschrift (z.B. die Verwendung von Pseudonymen) derart, dass die Einzelangaben über persönliche oder sachliche Verhältnisse ohne Kenntnis oder Nutzung der Zuordnungsvorschrift nicht mehr einer natürlichen Person zugeordnet werden können.

Pseudomisierung

□

Bei jedem Internetzugriff (z.B. beim Surfen) fallen eine Vielzahl von Daten über persönliche und sachliche Verhältnisse des Nutzers an, obwohl diese häufig für die Abwicklung der Dienstprotokolle nicht erforderlich sind. Beispiele dafür sind die IP-Adresse des Nutzers, seine Herkunftsadresse, die URL der zuvor geladenen Web-Seite oder auch Datum und Uhrzeit des Zugriffs. Diese Daten können sowohl von Angreifern (Dritten), die die Datenleitungen abhören, als auch von Dienst-Anbietern dazu verwendet werden, unautorisiert Profile über Nutzer zu erstellen. Anonymisierungsdienste versuchen, durch Kombinationen aus Vermeidungs- (z.B. Daten unterdrücken) und Verschleierungstechniken (z.B. Ersetzen durch Standardmuster, Verschlüsseln) eine Anonymisierung oder Pseudomisierung zu erzielen. Pseudonymität erzielt man durch die Einführung von Pseudony-

Anonymisierungsdienste

men, so dass die Identität eines Subjekts zwar einem vertrauenswürdigen Dritten bekannt ist, nicht jedoch jedem Kommunikationspartner. Im engeren Sinn zielen Anonymisierungstechniken darauf ab, Aufenthaltsorte und Kommunikationsbeziehungen so zu verschleiern, dass keine Bewegungs-, Kommunikations- oder Zugriffsprofile einzelner Benutzer durch unautorisierte Dritte erstellt werden können.

Privacy

Die oben angesprochenen Möglichkeiten, anhand von Verkehrs- und Aufenthaltsdaten Profile zu erstellen, führen zu einer Bedrohung der Privatsphäre (engl. *privacy*) des Nutzers. Problematisch ist, dass der Nutzer i.d.R. keinen Einfluss darauf hat, ob und in welchem Umfang derartige sensible Daten erhoben und gespeichert werden. So ist es beispielsweise beim Versenden einer EMail durchaus wünschenswert, die eigene EMail-Adresse anzugeben, während man aber dem Empfänger oder unbeteiligten Dritten keine weitere Informationen, wie z.B. die URL der zuvor genutzten WWW-Seite zukommen lassen möchte. Das Recht auf die informationelle Selbstbestimmung wurde bereits 1983 vom Bundesverfassungsgericht im Volkszählungsurteil verankert. Das Grundrecht gewährleistet die Befugnis des Einzelnen, grundsätzlich selbst über die Preisgabe und Verwendung seiner persönlichen Daten zu bestimmen. Einschränkungen dieses Rechts auf informationelle Selbstbestimmung sind nur im überwiegenden Allgemeininteresse zulässig. Mit dem informationellen Selbstbestimmungsrecht hat das Bundesverfassungsgericht auch den Anspruch auf Anonymisierung anerkannt (BVerfGE 65, 1, 49).

informationelle Selbstbestimmung

Mit der zunehmenden Vernetzung und den Entwicklungen im Bereich des so genannten ubiquitären Computing[1] nehmen die Bedrohungen der Privatheit sehr stark zu. Insbesondere der Einsatz und die Nutzung von RFID (Radio Frequency Identification) Chips führen zu einem Anstieg der Bedrohungen für die Privatsphäre. Passive und aktive RFID-Chips werden vermehrt in Objekte unterschiedlichster Art (u.a. Container, Automobile, Kleidungsstücke, Reisepässe, Tiere) eingebracht, um den Aufenthaltsort der Objekte (Tracking und Tracing) effizient ermitteln bzw. der Zustand der Objekte (z.B. wann wurde das Objekt zuletzt gewartet) einfach prüfen zu können. Dazu speichern die Chips Daten über das Objekt, wozu insbesondere die Identität des Objekts gehört. Diese Daten können über drahtlose Kommunikationsverbindungen mittels Lesegeräten ausgelesen werden, wobei die Daten heute in der Regel im Klartext zwischen dem RFID-Chip und dem Lesegerät übertragen werden. Zu Zeit besteht noch ein erheblicher Forschungsbedarf, um die Bedrohungen, die mit der ubiquitären Vernetzung einhergehen, mit geeigneten Maßnahmen wirkungsvoll abzuwehren.

RFID

[1] Synonym werden auch häufig die Begriffe Pervasive Computing oder Ambient Intelligence verwendet.

Vertrauen

Vertrauen

In den letzten Jahren wird der Begriff des Vertrauens (engl. *trust*) sehr häufig verwendet. Im Gegensatz zu den eingeführten Schutzzielen, beschreibt der Vertrauensbegriff keine Sicherheitseigenschaft eines Systems. Laut der Definition des Dudens bedeutet Vertrauen, das feste Überzeugtsein von der Verlässlichkeit und Zuverlässigkeit einer Person oder Sache. Ob diese Person oder Sache dieses Vertrauens auch würdig ist, also vertrauenswürdig ist, wird mit dem Vertrauensbegriff nicht erfasst. Vertrauen gemäß der Duden Definition ist also eher ein Gefühl, denn eine Eigenschaft, die man auch beweisen und nachweisen kann. Überträgt man den Vertrauensbegriff auf IT-Systeme, so könnte man definieren, dass ein Trusted System ein System ist, dem man vertraut, dass es spezifizierte Dienste, bzw. eine spezifizierte Funktionalität auch beim Auftreten von Störungen oder Fehlern korrekt und verlässlich erbringt. Sicherheitstechnologien, wie sie in diesem Buch eingehend behandelt werden, können dazu verwendet werden, um Vertrauen in IT-Systeme aufzubauen. Beispiele hierfür sind die Verwendung von Identitätsnachweisen, um das Vertrauen zu erhöhen, dass Dienste nur von authentischen und autorisierten Akteuren genutzt werden.

vertrauenswürdig

Eine Präzisierung des Begriffs *vertrauenswürdig* bzw. der *Vertrauenswürdigkeit* ist schwierig. In dem Buch *The Trusted Advisor* [53], das jedoch nicht für den IT-Bereich, sondern allgemein für Marketing-Interessierte geschrieben ist, findet man hierfür eine interessante Formel. Die Formel lautet wie folgt:

$$\text{Vertrauenswürdigkeit} = \frac{\text{Glaubwürdigkeit} + \text{Zuverlässigkeit} + \text{Vertrautheit}}{\text{Selbstorientierung}}$$

Wendet man eine solche Formel auf den IT Bereich an, um die Vertrauenswürdigkeit von Diensteanbietern in einer offenen Dienst-orientierten Umgebung, oder in einer offenen Cloud zu bewerten, so könnte man zur Bewertung der Glaubwürdigkeit und Zuverlässigkeit eines Dienstes bzw. des Diensteanbieters Reputationssysteme verwenden, die Auskunft geben, ob der Dienst entsprechend seiner Funktionalität zuverlässig arbeitet. Vertrautheit ergibt sich durch die eigene Erfahrung mit dem Dienst oder dem Provider, der den Dienst anbietet, während die Selbstorientierung schwierig zu bewerten sein wird. Würde man die Attribute der Formel quantifizieren, so ist klar, dass ein hoher Wert bei der Selbstorientierung den Grad der Vertrauenswürdigkeit verringert. Das erscheint sehr einleuchtend, da eine hohe Selbstorientierung eines Diensteanbieters darauf schließen lässt, dass der Diensteanbieter sehr stark nur seinen eigenen Vorteil im Auge hat.

Obwohl in den heutigen IT-Systemen sehr häufig von deren Vertrauenswürdigkeit die Rede ist, werden wir im Folgenden diese Begrifflichkeiten aufgrund der mangelnden Präzision der Begriffsbildung nicht verwenden. Wir werden jedoch in Kapitel 11 im Zusammenhang mit den Techniken des Trusted Computing auf das Thema noch einmal eingehen.

1.3 Schwachstellen, Bedrohungen, Angriffe

Ein IT-System kann unterschiedliche Schwachstellen besitzen. Wird eine dieser Schwachstellen ausgenutzt, so kann es zu Beeinträchtigungen der Datenintegrität, Informationsvertraulichkeit oder auch Verfügbarkeit kommen.

Definition 1.10 (Schwachstelle und Verwundbarkeit)

Schwachstelle

Verwundbarkeit

Unter einer Schwachstelle (engl. *weakness)* verstehen wir eine Schwäche eines Systems oder einen Punkt, an dem das System verwundbar werden kann. Eine Verwundbarkeit (engl. *vulnerability)* ist eine Schwachstelle, über die die Sicherheitsdienste des Systems umgangen, getäuscht oder unautorisiert modifiziert werden können.

□

Beispielsweise zählt die Diebstahlgefahr zu den physischen Schwachstellen mobiler Geräte, während Naturkatastrophen wie Feuer, Hochwasser, Erdbeben, Blitzschlag oder auch Stromausfall den natürlichen Schwachstellen zuzurechnen sind. Weitere Schwachstellen können durch die unsachgemäße Nutzung des Systems, durch Softwarefehler oder auch durch unsichere Kommunikationsverbindungen entstehen. Typische softwarebedingte Schwachstellen, die häufig zu einer Verwundbarkeit werden, sind Programme, in denen aufgrund einer fehlenden Eingabeüberprüfung Pufferbereichsüberläufe (vgl. Abschnitt 2.2) nicht korrekt abgefangen werden.

Gefährdungsfaktor

Um den Schutzbedarf eines IT-Systems zu ermitteln, ist es notwendig, eine detaillierte Schwachstellenanalyse durchzuführen (vgl. Abschnitt 4.4). Zur Ermittlung der Gefährdungslage und möglicher Schwachstellen sollte man sich zunächst einen Überblick über die vorhandenen Gefährdungsfaktoren verschaffen. Eine mögliche Klassifikation solcher Faktoren, wie sie beispielsweise in den BSI-Lehrbriefen zur IT-Sicherheit vorgeschlagen wird [37], ist der Abbildung 1.2 zu entnehmen.

1.3.1 Bedrohungen

Ein IT-System ist vielfältigen Bedrohungen ausgesetzt, die mögliche Gefährdungen für das System darstellen.

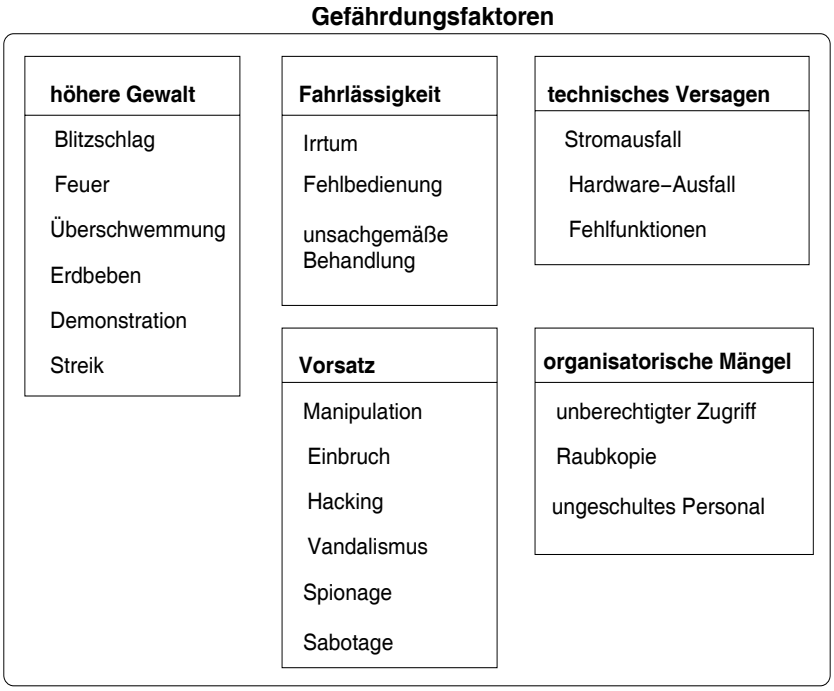

Abbildung 1.2: Klassifikation von Gefährdungsfaktoren

Definition 1.11 (Bedrohung)

Bedrohung

Eine Bedrohung (engl. *threat*) des Systems zielt darauf ab, eine oder mehrere Schwachstellen oder Verwundbarkeiten auszunutzen, um einen Verlust der Datenintegrität, der Informationsvertraulichkeit oder der Verfügbarkeit zu erreichen, oder um die Authentizität von Subjekten zu gefährden.

□

Gewichtung

Abhängig von der Funktionalität und Einsatzumgebung des Systems besitzen Bedrohungen ein unterschiedliches Gewicht. Eine unerlaubte Informationsgewinnung stellt zum Beispiel für öffentliche Datenbanken keine schwerwiegende Bedrohung dar, so dass auf aufwändige Maßnahmen zu deren Abwehr verzichtet werden kann. Im Gegensatz dazu stellt die Gewinnung von Informationen über Kundendaten oder Daten aus Forschungslabors für Unternehmen eine ernsthafte Bedrohung dar, die abzuwehren ist. Zur Bestimmung der tatsächlichen Gefährdungslage muss deshalb zunächst das Risiko bestimmt werden, das mit den potentiellen Bedrohungen verknüpft ist. Dazu sind die zu schützenden Güter, die *assets* zu bewerten und es ist das Schadenspotential, das durch das Eintreten eines Schadensereignisses auftreten kann, zu bestimmen. Ferner ist abzuschätzen, wie hoch die Eintrittswahrscheinlichkeit für Bedrohungen anzusiedeln ist. Damit ergibt sich nachfolgende Festlegung.

Definition 1.12 (Risiko)

Risiko

Unter dem Risiko (engl. *risk*) einer Bedrohung verstehen wir die Wahrscheinlichkeit (oder relative Häufigkeit) des Eintritts eines Schadensereignisses und die Höhe des potentiellen Schadens, der dadurch hervorgerufen werden kann.

□

Das jeweilige Bedrohungsrisiko hängt in hohem Maß von den zu Grunde liegenden Angreifermodellen ab. In diesen Modellen versucht man, die potentiellen Angreifer nach ihren Fähigkeiten und Zielen zu klassifizieren (z.B. Kenntnisse, Ressourcen, Budget), um dann in Abhängigkeit von der Höhe der zu schützenden Werte zu einer aussagekräftigen Aussage hinsichtlich des Schutzbedarfs zu gelangen. Es ist die Aufgabe der Bedrohungs- und Risikoanalyse (vgl. Abschnitte 4.4 und 4.5) zu klären, welche tatsächlichen Bedrohungen für das System zu beachten sind und welche Bedeutung sie für das System besitzen, d.h. welches Risiko besteht. Dazu wird zum einen die Eintrittswahrscheinlichkeit für die Bedrohungen abgeschätzt und zum anderen wird das Schadenspotential ermittelt, das ein erfolgreicher Angriff birgt. Um das Schadenspotential zu bestimmen, ist es notwendig, die sicherheitsrelevanten Objekte, also die zu schützenden Werte zu erfassen und deren Bedeutung innerhalb des Systems, des Unternehmens oder auch der Gesellschaft zu quantifizieren. Dies ist sehr schwierig und erfordert hohe Kompetenz und großes Fachwissen sowohl über organisatorische Strukturen als auch über betriebliche Abläufe (Prozesse) und natürlich auch detaillierte Kenntnisse über die technischen Komponenten und deren Sicherheitsprobleme. Abbildung 1.3 visualisiert den Zusammenhang zwischen Schwachstellen, Bedrohungen und Risiken.

Analyse

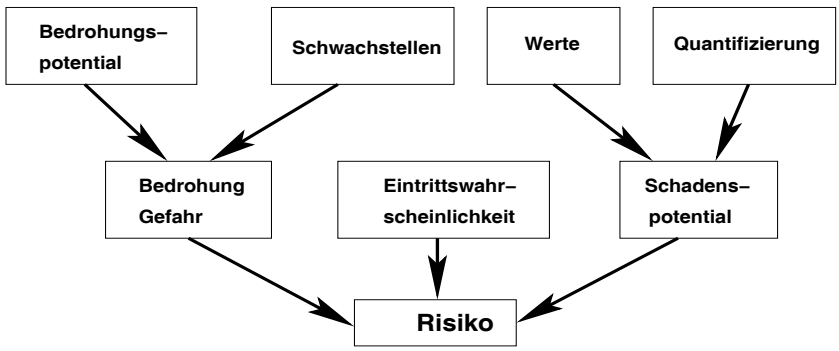

Abbildung 1.3: Zusammenhang zwischen Schwachstellen, Bedrohungen und Risiken

1.3.2 Angriffs- und Angreifer-Typen

Bedrohungen ergeben sich aus passiven und aktiven Angriffen auf das System.

Definition 1.13 (Angriff)

Angriff

Unter einem Angriff (engl. *attack*) verstehen wir einen nicht autorisierten Zugriff bzw. einen nicht autorisierten Zugriffsversuch auf das System. Wir unterscheiden passive und aktive Angriffe. Passive Angriffe betreffen die unautorisierte Informationsgewinnung und zielen auf den Verlust der Vertraulichkeit ab. Aktive Angriffe betreffen die unautorisierte Modifikation von Datenobjekten und richten sich somit gegen die Datenintegrität oder Verfügbarkeit eines IT-Systems.

□

passiver Angriff

Beispiele für passive Angriffe sind das Abhören von Datenleitungen in vernetzten Systemen (engl. *eavesdropping*) oder das unautorisierte Lesen von Daten aus Dateien. Zurzeit zählen Angriffe, durch die Passwörter ausgespäht werden, die so genannten Sniffer-Angriffe, zu den häufigsten passiven Angriffen im Internet.

Sniffer

aktiver Angriff

Beispiele für aktive Angriffe sind das Verändern oder Entfernen von Daten-Paketen aus einem Strom von übertragenen Nachrichten, das Wiedereinspielen von Nachrichten oder das unautorisierte Schreiben in Dateien. Zu den aktiven Angriffen zählen auch zwei spezielle Klassen von Attacken, die besonders häufig durchgeführt werden. Die erste Klasse umfasst Maskierungsangriffe, auch als Spoofing-Angriffe bekannt, die zum Ziel haben, eine falsche Identität vorzuspiegeln. Mittels solcher Maskierungsangriffe wird zum Beispiel versucht, eine falsche Absenderadresse in einer E-Mail anzugeben (E-Mail Address Spoofing). Ein mögliches Angriffsziel könnte hierbei darin bestehen, den Empfänger der Mail zur Preisgabe sensibler Informationen zu veranlassen. Eine weitere Variante solcher Spoofing-Angriffe ist der Versuch, die Identität eines Serverrechners anzunehmen (DNS-Name Spoofing oder IP-Address Spoofing (siehe auch Seite 168), um Anfragen von Clients selber z.B. mit gefälschten Daten zu beantworten bzw. zu bearbeiten. Die zweite spezifische Klasse umfasst Angriffe, die darauf abzielen, die Verfügbarkeit von Systemkomponenten oder -diensten infrage zu stellen. Man spricht hier von so genannten Denial-of-Service oder auch Resource Clogging Angriffen. Beispiele dafür sind das Überschwemmen von Rechnernetzen mit Nachrichten, so dass der Netzverkehr erheblich beeinträchtigt wird, oder das Blockieren eines Service-Ports eines Web-Service Anbieters (z.B. amazon.com) durch eine synthetisch erzeugte Überlast an Anfragenachrichten.

Spoofing

Denial-of-Service

Abwehr von Angriffen

Passive Angriffe können nur schwer verhindert werden, da auch die bereits angesprochenen verdeckten Kanäle Ansatzpunkte für passive Angriffe bieten und diese Kanäle nicht vollständig auszuschließen sind. Durch geeignete Maßnahmen kann man höchstens dafür sorgen, dass die Bandbreite dieser Kanäle minimiert wird. Mit kryptografischen Verfahren (vgl. Kapitel 7) stehen jedoch wirksame Mechanismen zur Verfügung, um solche passiven Angriffe wirkungslos zu machen, durch die Datenleitungen abgehört oder Speicherbereiche ausgelesen werden. Diese Mechanismen bieten auch einen gewissen Schutz gegen die angesprochenen Sniffer-Angriffe. Werden sensible Daten, wie beispielsweise Passwörter, verschlüsselt übertragen, so können sie mittels Sniffer-Programmen nicht mehr ausgespäht werden. Verschlüsselungstechniken bieten aber keinen Schutz dagegen, verschlüsselte Passwörter zu kopieren und diese dann in der verschlüsselten Form als Authentifikationsmerkmal vorzuweisen. Zur Abwehr solcher Angriffe, die zur Klasse der Maskierungsangriffe gehören, sind weitere Maßnahmen (vgl. Kapitel 10) notwendig.

Verschlüsseln

minimale Rechte

Aktive Angriffe auf die Datenintegrität können durch eine Beschränkung von Rechten zur Durchführung von Schreibzugriffen verhindert oder zumindest begrenzt werden. Hierbei hilft der Einsatz von Sicherheitsmodellen (vgl. Kapitel 6), die es erlauben, von Realisierungsdetails zu abstrahieren und sich auf das Wesentliche, hier die Modellierung und Vergabe von Schreibberechtigungen, zu konzentrieren. Zusätzlich zu einer a priori Eingrenzung sind Mechanismen und Verfahren notwendig, um aktive Angriffe zu erkennen und wirkungslos zu machen. Das Erkennen aktiver Angriffe ist allein durch den Einsatz kryptografischer Verfahren nicht immer möglich, sondern erfordert zusätzliche Mechanismen, wie die Einführung von Sequenznummern. Damit ist man in der Lage zu kontrollieren, ob zusätzliche Daten in einen Datenstrom eingeschleust oder Daten entfernt wurden.

Monitoring

Die Abwehr von Angriffen auf die Verfügbarkeit ist schwierig. Eine mögliche Abwehrmaßnahme ist die oben bereits angesprochene Vergabe von Quoten zur Ressourcennutzung. Eine weitere Maßnahme besteht im intensiven Beobachten (engl. *monitoring*) der gefährdeten Ressourcen, um eine verdächtige Überlastsituation frühzeitig zu erkennen und durch gezielte Eingriffe zu beheben. Im Bereich der Rechnernetze stehen entsprechende Werkzeuge zur Analyse des Netzverkehrs zur Verfügung.

Grauzone

Die Vermeidung oder Verhinderung von Angriffen wird in vielen Fällen dadurch erschwert, dass sie auch aus unabsichtlichem Fehlverhalten von Benutzern resultieren, d.h. insbesondere durch fehlerhafte Programmierung hervorgerufen werden können. Vielfach ist es kaum nachweisbar, ob

ein Fehlverhalten auf Unachtsamkeit oder auf Absicht zurückzuführen ist. Beispiel 1.2 verdeutlicht diese Grauzone. Die Behandlung von Programmierfehlern und Techniken zum sicheren Programmieren fallen in den Bereich der Software Sicherheit bzw. in den Bereich Safety und werden hier nicht näher behandelt.

Beispiel 1.2 (Bug oder Angriff?)

Ein, wenn auch noch immer etwas umstrittenes Beispiel[2] dafür, dass es vielfach kaum möglich ist, zwischen absichtlichen Angriffen und unabsichtlichen, durch Programmierfehler entstandenen Attacken zu unterscheiden, war ein Programmierfehler, der in einem FORTRAN-Programm auftrat und zum Verlust einer amerikanischen Mariner 1 Venus Sonde führte. Auslöser soll eine absichtlich oder unabsichtlich fehlerhaft programmierte `DO`-Schleife gewesen sein. Anstatt des korrekten Codes

Mariner 1 Sonde

```
DO 20 I = 1,100
```

soll das Programm die Codezeile

```
DO 20 I = 1.100
```

enthalten haben. Die derart abgeänderte Codezeile hätte zur Folge, dass `DO20I` als eine implizit deklarierte Variable[3] interpretiert und eine Zuweisung an diese Variable durchgeführt würde, anstatt eine DO-Schleife zu durchlaufen. Es blieb bis heute unklar, ob es sich tatsächlich um einen Angriff zur Zerstörung der Sonde gehandelt hat, oder ob lediglich ein Programmierfehler bzw. ein schlichter Schreibfehler (Punkt statt Komma) vorlag.

▲

Auch wenn in dem Mariner 1 Fall die genaue Ursache des Problems nach wie vor debattiert wird, so ist dennoch unbestritten, dass ein einfacher Zeichenfehler in der Kontrollsoftware die Ursache für ein schwerwiegendes Problem war. Das Beispiel lehrt aber, dass Funktions- und Qualitätstests von Software unerlässlich sind, um Sicherheitsverletzungen, die aus fehlerhafter Programmierung resultieren können, frühzeitig zu erkennen. Das bedeutet, dass zur methodischen Konstruktion von sicheren Software-Systemen sowohl ingenieurmäßige Methoden und Vorgehensmodelle als auch geeignete Test-Werkzeuge notwendig sind, um komplexe Software von hoher Qualität zu erstellen und diese Qualität nachhaltig, also auch im laufenden Betrieb, garantieren zu können.

[2] siehe u. a. http://catless.ncl.ac.uk/Risks/5.66.html

[3] FORTRAN ignoriert Zwischenräume und erlaubt implizit deklarierte Variablen.

Angreifer-Typen

Angriffsursachen

Unterschiedliche Studien (u.a. [87]) verdeutlichen, dass nach wie vor ein großer Teil aller in Unternehmen, Firmen und Behörden bekannt gewordenen Angriffe durch interne Mitarbeiter, den Innentätern, erfolgen. Durch die zunehmende Vernetzung und Öffnung von Systemen nimmt jedoch die Zahl der externen Angriffe ganz erheblich zu und stellt insbesondere durch die zunehmende Kriminalisierung der Angriffe eine sehr große Bedrohung dar.

Hacker

Unter einem Hacker versteht man einen in der Regel technisch sehr versierten Angreifer, dessen Ziel es ist, Schwachstellen und Verwundbarkeiten in IT-Systemen aufzudecken und Angriffe, so genannte Exploits, zu entwickeln, um damit diese Schwachstellen auszunutzen. Hacker wenden sich mit diesen Exploits in der Regel an die Öffentlichkeit, um auf Schwachstellen aufmerksam zu machen. Sie verfolgen meist nicht das Ziel, aus der Ausbeutung von Schwachstellen persönliche Vorteile, wie finanzielle Gewinne oder bewusste wirtschaftliche Schäden Dritter, zu ziehen. Man spricht hier auch gerne von der Hacker-Ethik. Man sollte sich jedoch klar darüber sein, dass Hacker zur Durchführung ihrer Angriffe häufig illegale Wege einschlagen, sich also außerhalb der gesetzlichen Regelungen bewegen.

Cracker

Ein Cracker ist ebenfalls ein technisch sehr versierter Angreifer, der jedoch im Unterschied zu einem Hacker Angriff gezielt zu seinem eigenen Vorteil oder zum Nachteil eines Dritten durchführt. Von der Gruppe der Cracker geht somit ein weit größeres Schadensrisiko aus als von Hackern, so dass Unternehmen, die lukrative Angriffsziele darstellen, einen erheblichen Aufwand leisten sollten, um Cracker-Angriffe wirkungsvoll abzuwehren oder zumindest deren Schadensausmaß zu begrenzen.

Skript Kiddie

Den zuvor genannten, technisch versierten Angreifern steht mit den Skript Kiddies eine stetig wachsende Zahl von Internet-Nutzern gegenüber, die zwar über viel Zeit, aber nicht notwendigerweise über vertieftes technisches Know-how verfügen. Diese Gruppe nutzt die frei verfügbaren Exploits, um ihrerseits Angriffe durchzuführen. Obwohl diese Gruppe meist eher von Motiven wie dem Spieltrieb und der Neugierde denn durch die Absicht, Dritte vorsätzlich zu schädigen, getrieben ist, bedeutet die Leichtigkeit, mit der Skript Kiddies vorgefertigte Exploits (s.u.) durchführen können, eine erhebliche Bedrohung für heutige und erst recht zukünftige IT-Systeme. Zu beachten ist aber, dass zu den veröffentlichten Exploits meist sehr schnell Patches zum Schließen der ausgebeuteten Schwachstellen zur Verfügung gestellt werden (z.B. unter http://www.securityfocus.com). Somit lassen sich Nachahmungstäter effektiv abwehren, falls die zuständigen Sicherheitsadministratoren die einschlägigen Mailinglisten und Diskussionsforen überwachen und Patches möglichst zeitnah einspielen.

Wirtschafts-spionage

In jüngster Zeit mehren sich auch Lauschangriffe auf Datenleitungen und ISDN-Verbindungen zum Zwecke der Wirtschaftsspionage oder als Folge geheimdienstlicher Tätigkeiten. So verlieren heute nach einer Schätzung des Bundesinnenministeriums deutsche Unternehmen pro Jahr noch immer 20 Milliarden Euro durch Wirtschaftsspionage[4].

Unter dem Codenamen Echelon betreibt die NSA (National Security Agency) in den USA angeblich schon seit Anfang der 80er Jahre eine flächendeckende Überwachung aller Telefonate, Faxe, E-Mails und Telexe, die über internationale Telekommunikationssatelliten, regionale Satelliten sowie über Kabel und Funktürme gesendet werden. Die aufgezeichneten Daten werden mittels spezieller Suchprogramme nach verdächtigen Begriffen abgesucht und beim Auffinden solcher Begriffe werden die betroffenen Behörden bzw. Stellen informiert. Eindrucksvolle Beispiele für bekannt gewordene Angriffe im Zusammenhang mit der Wirtschaftsspionage, durch die einzelne Unternehmen beträchtliche finanzielle Einbußen hinnehmen mussten[5], sind der Verlust eines 3 Milliarden Euro Auftrags von Airbus Industries, da die NSA den Datenaustausch mit dem potentiellen Auftraggeber Saudi Arabien abgehört und die Informationen der heimischen Industrie weitergeleitet haben soll. Zu nennen ist hier auch das Scheitern des Verkaufs der ICE-Technologie von Siemens nach Süd-Korea. Hier soll der französische Geheimdienst die Nachrichten abgefangen und an die heimische Industrie weitergereicht haben, wodurch Siemens ein Verlust in der Größenordnung von über 2 Milliarden Euro entstanden sein soll.

Kriminelle

Neben der Wirtschaftskriminalität durch Industrie-Spionage ist aber auch ein Anstieg an anderen kriminellen Delikten unter Nutzung der vernetzten IT-Systeme zu beobachten. So verzeichnete im Juni 2005 der IBM Global Business Security Index für das erste Halbjahr 2005 einen Anstieg um über 50 Prozent bei den kriminellen Attacken. Der Index fasst Informationen von weltweit 2700 IBM Information Security Spezialisten und einer halben Million überwachter Geräte zusammen. Nach den heutigen Erkenntnissen greifen Kriminelle zunehmend bestimmte Zielgruppen an, wie Regierungen oder Firmen aus dem Finanzsektor, und verursachen einen hohen individuellen Schaden. Neben unterschiedlichen Formen der Erpressung steht zurzeit die profitorientierte Web-Kriminalität an der Spitze der Skala. Die E-Mail bzw. auch Browser-basierten Phishing-Angriffe[6], bei denen der Benutzer durch gezielte Täuschungsversuche dazu verleitet wird, sein Passwort preiszugeben, haben hieran nach wie vor einen erheblichen Anteil.

---

[4] http://www.computerwoche.de/knowledge_center/it_security/1852724/

[5] siehe u.a. http://www.heise.de/security/meldung/Innenministerium-warnt-vor-zunehmen der-Wirtschaftsspionage-1223563.html

[6] Phishing ist aus den Begriffen Password und Fishing entstanden.

Phishing-Angriffe sind eine spezielle Ausprägung von Identitätsdiebstählen, deren Zahl stetig zunimmt. Nach einem Ergebnis einer Umfrage des Branchenverbandes Computing Technology Industry Association (CompTIA) im Jahre 2005 waren von fast 500 befragten Organisationen 57 Prozent Opfer von Browser-basierten Attacken. Im Vorjahr waren es nur 37 Prozent und zwei Jahre zuvor sogar nur 25 Prozent. Von Kosten in Höhe von 400 Milliarden Dollar allein in den USA im Jahr 2006, die durch Computer-Kriminalität verursacht wurden, spricht die US-Polizeibehörde FBI in Washington.

Bot-Netz

Kriminelle bedienen sich für ihre Aktivitäten zunehmend des Know-hows von technisch versierten Angreifern. Zu nennen ist hier insbesondere der Aufbau von so genannten Bot-Netzen durch solche Angreifer. Hierfür werden viele (in der Regel mehrere tausend) PCs so präpariert, dass von diesen präparierten PCs gezielt Angriffe auf ein Opfersystem gefahren werden können (vgl. Seite 78). Die dafür erforderliche Schadsoftware wird in der Regel über Trojanische Pferde (vgl. Kapitel 2) unbemerkt vom PC-Besitzer auf dessen PC eingeschleust. Die Ausführung der Schadsoftware wird entfernt durch den Angreifer kontrolliert und initiiert, um z.B. einen Denial-of-Service (DoS) Angriff auf ein bestimmtes Opfersystem durchzuführen. Das Bot-Netz wird von den Angreifern gerne an Kriminelle vermietet, die das vorbereitete Netz u.a. für Erpressungen gegen den Betreiber des Opfersystems nutzen. In einem solchen Bot-Netz-Szenario sind dann natürlich die Besitzer der missbrauchten und präparierten PCs nicht nur Opfer einer Schadsoftware-Attacke sondern unwissentlich auch selber Angreifer, wenn ihr PC an einer entsprechenden DoS Attacke beteiligt ist.

Erpressung

Die steigende Kriminalisierung der Internet-basierten Angriffe spiegelt sich auch in der Zunahme an erpresserischer Malware wider, die unter dem Kunstnamen Ransomware bekannt geworden sind. Hierbei handelt es sich um Schadsoftware, die auf fremden Rechnern eindringt und Daten auf der lokalen Festplatte des fremden Rechners verschlüsselt, so dass diese Daten für den Nutzer nicht mehr zugreifbar sind. Für die Entschlüsselung der Daten fordert der Angreifer üblicherweise einen Geldbetrag, der über ein Online-Zahlungssystem wie PayPal zu entrichten ist. Die Bezeichnung der Ransomware ergibt sich aus der Kombination aus dem Begriff Malware sowie dem englischen Begriff für Lösegeld (ransom). Eine bekannte Variante dieser Schadsoftware war der Virus Gpcode, der sich im Sommer 2006 verbreitete, bzw. seine neuen Varianten aus dem Jahr 2007. Gpcode wurde über SPAM-Mails verbreitet. Der Virus verschlüsselt Daten der lokalen Festplatte, die eine spezifische Endung haben, mittels RSA. Die neueren Varianten des Gpcode-Virus verwenden häufig keine RSA-Verschlüsselung mehr, sondern legen Daten von der lokalen Festplatte in einem zip-Archiv ab, das mit einem starken Passwort gesichert wird. In analoger Weise wird

dann ein Geldbetrag dafür gefordert, dass der Angreifer dem Nutzer dieses Passwort nennt, um ihm seine eigenen Daten wieder zugänglich zu machen.

BSI-Lageberichte

Die Lageberichte[7] zur IT-Sicherheit in Deutschland des Bundesamts für Sicherheit in der Informationstechnik (BSI) bestätigt diese Trends zur Kommerzialisierung und Professionalisierung der Internetkriminalität. Die Berichte verzeichnen einen nach wie vor deutlichen Anstieg von Angriffen, die Schadsoftware (engl. *malware*) mittels Viren, Würmer und Trojanische Pferde verbreiten. Auffallend ist, dass klassische Viren-Angriffe (vgl. Kapitel 2) zunehmend durch die gezielte Verbreitung Trojanischer Pferd-Programme ersetzt werden. Ein Hauptziel derartiger Trojaner besteht darin, unbemerkt Spyware Programme auf Rechnern zu installieren. Hierbei handelt es sich um Software, die gezielt auf fremden Rechnern Informationen sammelt und dem Angreifer zugänglich macht. Beispiele für derart ausgespähte Informationen sind auf der Festplatte gespeicherte Passwörter und Zugangscodes, oder aber Passwörter und Codes, die über die Tastatur eingegebene werden. Sensitive Dokumente, E-Mails oder aber auch Cookies, die Zugangsinformationen für Server im Web beinhalten, sind weitere Beispiele von unberechtigt ausgespähten Daten. Da ein großer Teil dieser ausgespähten Daten zur Identifizierung von Nutzern im System oder im Internet (u.a. Zugangscode zu Banken und Portalen) dienen, sind sie eine der Hauptursachen für die noch immer wachsende Zahl der Identitätsdiebstähle. Weiterhin verzeichnen die Lageberichte, dass Schadprogramme immer effektiver arbeiten, die Zahl der Denial-of-Service-Angriffen zunimmt und die Sicherheitslücken in Betriebssystemen und Anwendungen immer schneller ausgenutzt werden. Dies wird beispielsweise eindrucksvoll untermauert von der rasanten Ausbreitung des Conficker-Wurms Ende 2008 bzw. Anfang 2009. Nach Schätzungen von Antivirenhersteller hatte der Windows-Wurm bereits Anfang 2009 rund 2,5 Millionen PCs infiziert. Gesamtwirtschaftliche Kosten, die durch den Befall des Wurms entstanden sind, lassen sich nur schwer quantifizieren, jedoch halten Institute wie das Cyber Security Institute[8] Schäden in der Größenordnung von bis zu 200 Millionen US Dollar für durchaus realistisch. Auch der Bericht aus dem Jahr 2011[9] bestätigt diese generellen Trends. Zusätzlich wird darauf verwiesen, dass im Bereich der Cyber-Kriminalität eine zunehmende Kommerzialisierung zu verzeichnen ist. So werden große Botnetze von z.B. 10.000 PCs zum Anmieten für 200 US Dollar pro Tag auf dem Markt angeboten. Mit einem solchen Botnetz können z.B. SPAM-Angriffe durchgeführt werden, die sehr rentabel

[7] vgl. https://www.bsi.bund.de/ContentBSI/Publikationen/Lageberichte/bsi-lageberichte.html

[8] http://cybersecureinstitute.com

[9] vgl. https://www.bsi.bund.de/SharedDocs/Downloads/DE/BSI/Publikationen/Lageberichte/Lagebericht2011_nbf.pdf

für die Betreiber des Botnetzes sein können. Ein weiterer sich fortsetzender Trend, der in dem Bericht angesprochen wird, ist die deutliche Zunahme von Schwachstellen bei Anwendungsprogrammen und bei Software von Drittanbietern, wie Adobe Flash Player oder Mozilla Firefox. Erhebliche Zuwachsraten an Sicherheitsproblemen wird im Bereich der Smartphones mit den mobilen Apps (u.a. Android und iPhone) verzeichnet. In Zukunft werden insbesondere im Cloud-Computing ganz erhebliche Gefährdungen gesehen. Positiv hat sich den Berichten zu Folge die Bewusstseinsbildung entwickelt. Das Sicherheitsbewusstsein der Anwender ist gestiegen, Betriebssystem-Updates werden häufiger durchgeführt und IT-Sicherheitstechniken deutliche häufiger angewendet.

Mitarbeiter

Die meisten der durch interne Mitarbeiter verursachten Angriffe gehen auf mangelhafte Kenntnisse sowohl der Systemgegebenheiten als auch der zur Verfügung stehenden Sicherheitsmechanismen und deren Bedeutung zurück. Weitere Hauptquellen für Angriffe sind Nachlässigkeiten im Umgang mit dem System und den zu verwaltenden sensiblen Informationen. Auch ein mangelhaftes Problembewusstsein, sowohl bei den Mitarbeitern als auch im mittleren und im Top-Management, wird in den entsprechenden Studien wiederholt angeprangert. Mit Maßnahmen des ganzheitlichen Sicherheitsmanagements könnten derartige organisatorische sowie technische Sicherheitsmängel behoben werden. Bei einem ganzheitlichen Ansatz werden sowohl organisatorische Strukturen als auch die Geschäftsprozesse und die technischen Komponenten zusammen betrachtet, um Brüche zwischen diesen verschiedenen Ebenen zu vermeiden.

Social Engineering

Schließlich ist auch noch auf Angriffe hinzuweisen, die nicht technischer Natur sind. Man spricht vom Social Engineering oder Social Hacking. Hierbei versucht ein Angreifer sein Opfer dazu zu bringen, dass es unabsichtlich oder absichtlich im guten Glauben sensitive Informationen an den Angreifer preisgibt. Ein sehr beliebter Ansatz ist es, sich gegenüber Mitarbeitern als Systemadministrator auszugeben (z.B. über einen Telefonanruf) und um die Angabe des Benutzer-Passworts zu bitten, um angeblich dringend notwendige administrative Arbeiten durchführen zu können.

Entwicklung

Für die Zukunft wird ein weiteres Ansteigen der Identitätsdiebstähle sowie eine Zunahme der Bedrohungen durch Schadsoftware und Spyware, insbesondere durch Trojanische Pferde erwartet. Ferner wird mit einem weiteren Anstieg von solchen Angriffen gerechnet, die durch Unkenntnis und Nachlässigkeit hervorgerufen werden, so dass der Bedarf an wirksamen Sicherheitsmechanismen und -diensten wächst und die Schulung von Mitarbeitern bzw. die fundierte Ausbildung auf dem Gebiet der Systemsicherheit unerlässlich ist. Ein weiterer Anstieg der Angriffe im Bereich der Computer-

Cyber-Crime

kriminalität bis hin zu Cyber-Terrorismus ist ebenfalls zu erwarten.

Rootkit

Rootkit

Ein besonderes Gefahrenpotential bergen die so genannten Rootkits oder Root-Exploits. Unter Rootkits verstehen wir eine Klasse von Angriffen auf IT-Systeme, die schnell, vollautomatisch und auch ohne Expertenwissen mit hoher Wahrscheinlichkeit erfolgreich unter Nutzung frei verfügbarer Angriffs-Software durchgeführt werden. Die Angriffe haben zum Ziel, auf dem angegriffenen Rechner Administrator-Rechte (Root-Rechte) zu erlangen. D.h. im Erfolgsfall erhält der Angreifer die volle Systemkontrolle. Angriffs-Schemata, die dies ermöglichen, nennt man auch häufig Root-Exploits. Ein Rootkit-Angriff läuft meist nach einem ähnlichen Muster ab:

Ablauf

- *Schwachstellensuche*
  Zunächst erfolgt eine systematische Schwachstellenanalyse auf der Basis von Expertenwissen, das in öffentlich zugänglichen Datenbanken verfügbar ist. Diese Schwachstellen-Datenbanken liefern Informationen über gängige Betriebssysteme, systemnahe Dienste und deren Verwundbarkeiten. Der Exploit sucht deshalb nach Rechnern, die bekannte Schwachstellen aufweisen. Dazu werden mittels Portscans die Rechner eines Netzsegments nach offenen Ports, z.B. FTP, HTTP, maild, inetd etc. abgesucht.
- Die gefundenen Rechner und Informationen werden mit den Daten aus der Schwachstellen-Datenbank abgeglichen und es erfolgt i.d.R. die Auswahl des am einfachsten angreifbaren Dienstes.
- Alternativ wird häufig direkt nach Schwachstellen in den Systemkonfigurationen gesucht, wie beispielsweise nach Kennungen, für die keine Passwörter eingetragen sind, oder nach Standardpasswörtern für Standardkennungen (z.B. Passwort Gast für die Kennung Gast).
- Ist eine Schwachstelle identifiziert, so wird versucht, den Angriffs-Code darüber in das System zu schleusen. Sehr häufig erfolgt das über den Weg des Buffer-Overflow-Angriffs (vgl. Abschnitt 2.2).
- *Verschleierung*
  Um seine Angriffe ungestört durchführen zu können, wird das Rootkit nun noch versuchen, seine Aktivitäten zu verschleiern. Dazu gehört das Säubern von Log-Dateien oder auch das Manipulieren von Dienstprogrammen, die Auskunft über Systemaktivitäten geben. Beispiele hierfür sind Programme wie *ps*, das die aktiven Prozesse des Systems aufführt, *netstat*, das die offenen Verbindungen auflistet oder *ls*, das Auskunft über das Dateisystem liefert.
- Der Angreifer hat ein Interesse daran, das von ihm angegriffene Zielsystem nachhaltig zu unterwandern, so dass er wiederholt, unentdeckt die Kontrolle über das System übernehmen kann. Hierzu wird gerne ausgenutzt, dass heutige Betriebssysteme das dynamische Nachladen von

Systemprogrammen ermöglichen (z.B. über das DLL-Konzept (Dynamic Link Library) unter Windows). Ein Rootkit versucht häufig, vorgefertigte, manipulierte Versionen von Standard-Systemdiensten, z.B. für *ls, find, netstat, ps, syslogd, login, ftp, sniffer* etc. nachzuladen, um beispielsweise weitere Passwörter von Benutzern auszuspähen.

Fazit

Rootkits nutzen i.d.R. bekannte Fehler der Systemsoftware aus, so dass Abwehrmaßnahmen auf der Hand liegen. So muss die Systemkonfiguration regelmäßig überprüft, auf neuesten Sicherheitsstand gebracht (Patches) und überwacht werden. Verdächtige Portscans sind zu verfolgen und mittels Einbruchserkennungs-Werkzeugen sollte versucht werden, auffälliges Verhalten, das auf einen Exploit schließen lässt, zu ermitteln. Natürlich sollten auch Kontroll- und Schutzmaßnahmen wie Verschlüsselungen, kryptografische Prüfsummen oder auch differenzierte Zugriffskontrollen (u.a. Root erhält nicht automatisch Rechte an allen Daten) eingeführt und angewandt werden.

1.3.3 Rechtliche Rahmenbedingungen

Gesetzeslage

Die Frage, inwieweit Angriffe von Crackern, Hackern und anderen Angreifertypen nach deutschem Recht strafbar sind, wird derzeit gerade neu überdacht und geordnet. So verbietet §202a des Strafgesetzbuches nur dann das Ausspähen von fremden Daten, wenn diese „gegen unberechtigten Zugang besonders gesichert sind." Am 1.2.2001 hat der Deutsche Bundestag das so genannte Zugangskontrolldiensteschutzgesetz (ZKDSG) verabschiedet. Das Ziel dieses Gesetzes ist es, die unberechtigte Nutzung kostenpflichtiger Angebote von Rundfunk- und Fernsehsendern sowie Tele- und Mediendiensten durch das Umgehen von technischen Vorsperr- und Verschlüsselungsverfahren zu unterbinden. Als Zugangskontrolldienste gelten sowohl Hardware-Lösungen wie Decoder und Smartcards als auch in Software implementierte Verschlüsselungsmechanismen.

Auswirkungen

Nach diesem Gesetz kann jeder mit einer Freiheitsstrafe bis zu einem Jahr belegt werden, der Einrichtungen für das Umgehen solcher Zugangskontrolldienste zu gewerbemäßigen Zwecken einführt, herstellt oder verbreitet. Ferner kann dann der gewerbliche Besitz, die Einrichtung, die Wartung und der Austausch der Cracker-Werkzeuge mit Geldbußen bis zu 50.000 Euro geahndet werden. Das Gesetz deckt insbesondere den Bereich des Pay-TV und der entgeltlichen Video-on-Demand Angebote ab. Nicht betroffen von der Regelung sind Techniken wie die elektronische Signatur, bei denen Daten aus Sicherheitsgründen verschlüsselt übertragen werden, oder auch der Kopierschutz bei Dateien. Mit diesem Gesetz werden aber nur gewerbsmäßig handelnde Personen erfasst, während Privatpersonen sich nicht

strafbar machen, wenn sie gefälschte Smartcards oder gehackte Software besitzen. Das ist durchaus eine sinnvolle Regelung, da ein gutgläubiger Nutzer relativ leicht auf ein illegales Betrugsangebot hereinfallen kann. Durch die gesetzliche Regelung soll der eindeutig Schuldige zur Verantwortung gezogen werden. Die private Nutzung einer illegalen Umgehungseinrichtung für Zugangskontrolldienste fällt in Deutschland unter den §265a des Strafgesetzbuches, der die „Erschleichung von Leistungen“ behandelt.

Phishing

Auch Phishing-Angriffe sind nach geltendem Recht strafbar, da über einen solchen Angriff Daten (hier Zugangsdaten) ausgespäht werden, mit dem Ziel, diese Daten zu missbrauchen, um beispielsweise beim Online-Banking unter der Identität des Opfers Geldtransaktionen zu Lasten des Opfers zu tätigen. Phishing-Angriffe können deshalb Straftatbestände des Ausspähens von Daten (Paragraph 202a StGB), des Betrugs/Computerbetrugs (§263/ §263a StGB), der Fälschung beweiserheblicher Daten (§269 StGB) und der unbefugten Datenerhebung und -verarbeitung (§44, 43 BDSG) betreffen.

Hacker-Paragraph

Im September 2006 hat das Bundeskabinett einen Regierungsentwurf eines Strafrechtsänderungsgesetzes zur Bekämpfung der Computerkriminalität[10] beschlossen, das den EU-Rahmenbeschluss über Angriffe auf Informationssysteme sowie das Europarat-Übereinkommen über Computerkriminalität in nationales Recht umsetzen sollte. Am 24.05.2007 hat der Bundestag diesen Entwurf trotz erheblicher Bedenken von Seiten von Sicherheitsexperten verabschiedet.
Die Änderungen umfassen eine Ergänzung des Paragraphen 202a Absatz 1, der nun wie folgt lautet: „(1) Wer unbefugt sich oder anderen Zugang zu Daten, die nicht für ihn bestimmt und die gegen unberechtigten Zugang besonders gesichert sind, unter Überwindung der Zugangssicherung verschafft, wird mit Freiheitsstrafen bis zu drei Jahren oder mit Geldstrafe bestraft.“

Erweitert wurde der Gesetzestext insbesondere um die Paragraphen 202b und 202c. Paragraph 202b behandelt den Straftatbestand des Abfangens von Daten: „Wer unbefugt sich oder einem anderen unter Anwendung von technischen Mitteln nicht für ihn bestimmte Daten (§202a Abs.2) aus einer nichtöffentlichen Datenübermittlung oder aus der elektromagnetischen Abstrahlung einer Datenverarbeitungsanlage verschafft, wird mit Freiheitsstrafe bis zu zwei Jahren oder mit Geldstrafe bestraft, wenn die Tat nicht in anderen Vorschriften mit schwererer Strafe bedroht ist.“

Der Paragraph 202c StGB betrifft das Vorbereiten des Ausspähens und Abfangens von Daten: „(1) Wer eine Straftat nach Paragraph 202a oder Paragraph 202b vorbereitet, indem er

[10] siehe http//www.buzer.de/gesetz/7846/index.htm

1. Passworte oder sonstige Sicherungscodes, die den Zugang zu Daten (§202a Abs. 2) ermöglichen, oder
2. Computerprogramme, deren Zweck die Begehung einer solchen Tat ist, herstellt, sich oder einem anderen verschafft, verkauft, einem anderen überlässt, verbreitet oder sonst zugänglich macht, wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft."

Kritik

Der Gesetzesentwurf wurde im Vorfeld heftig kritisiert. Hauptkritikpunkte betrafen den Paragraphen 202c des Strafgesetzbuchs(StGB), der den Einsatz und die Verbreitung von so genannten Hacker-Tools unter Strafe stellt. In einer Stellungnahme der Gesellschaft für Informatik (GI)[11] heißt es: „Problematisch ist die Einfügung des Paragraph 202c StGB, weil Programme und Tools nicht nach ihrer Einsatzart, sondern vielmehr nach ihrem Aufbau definiert werden. Eine Unterscheidung in Anwendungen, die zur Begehung von Straftaten und solche, die ausschließlich für legale Zwecke hergestellt werden, ist aber nicht möglich. Der gewählte Wortlaut führt zu einer Kriminalisierung der heute in allen Unternehmen, Behörden und von Privaten verwendeten Programme zur Aufdeckung von Sicherheitslücken in IT-Systemen. Derartige Programme und Tools sind zur Absicherung gegen Angriffe jedoch unverzichtbar (Penetration Testing)."

Zweckbestimmung

Prof. A. Rossnagel von der Universität Kassel führt in der GI-Stellungnahme hierzu weiter aus: „Es besteht die Gefahr, den bloßen Besitz und die informationstechnische Entwicklung von Tools zu bestrafen, die der Identifizierung von Sicherheitslücken dienen, zugleich aber wesensnotwendig auch zum Eindringen in Systeme verwendet werden können, [. . .]. Das angeblich einschränkende objektive Tatbestandsmerkmal der *Zweckbestimmung für eine Straftat* (§202c Abs. 1 Nr. 2 StGB-E) ist keines, weil Computerprogramme keinen Zweck haben. Selbst wenn der Entwickler einen bestimmten Zweck intendiert, können sie immer missbraucht werden. Die Begründung erkennt dies implizit an, wenn sie angibt, es reiche aus, wenn die objektive Zweckbestimmung *auch* die Begehung einer Straftat sei. Noch problematischer wird dies dadurch, dass der bedingte Vorsatz erfasst ist. Es ist eine Vielzahl von Fällen vorstellbar, in denen ein Betroffener bei einer an sich legitimen Handlung in Kauf nehmen wird, dass ein Passwort oder ein Computerprogramm auch anderweitig verwendet wird [. . .]. Bereits die Gefahr, wegen einer solchen Tätigkeit belangt zu werden, kann die Entwicklung und Verbesserung von Sicherheitstechnik behindern, Industrie und Bürgern wichtiger Selbstanalysemöglichkeiten berauben und so die IT-Sicherheit gefährden."

Kommentare

In der Drucksache 16/3656 des Bundestags[12] wird auf einige Kritikpunkte Bezug genommen. Dort ist unter anderem zu lesen: „Durch die objektive

[11] vgl. http://www.gi-ev.de/aktuelles/meldungsdetails/meldung/159/
[12] siehe http://dip.bundestag.de/btd/16/036/1603656.pdf

Beschränkung auf Computerprogramme, deren Zweck die Begehung einer Computerstraftat ist, wird bereits auf Tatbestandsebene sichergestellt, dass keine Computerprogramme erfasst werden, die beispielsweise der Überprüfung der Sicherheit oder Forschung in diesem Bereich dienen. Unter Strafe gestellt werden lediglich das Herstellen, Verschaffen, Verbreiten usw. solcher Programme, denen die illegale Verwendung immanent ist, die also nach Art und Weise des Aufbaus oder ihrer Beschaffenheit auf die Begehung von Computerstraftaten angelegt sind. Bei Programmen, deren funktionaler Zweck nicht eindeutig ein krimineller ist und die erst durch ihre Anwendung zu einem Tatwerkzeug eines Kriminellen oder zu einem legitimen Werkzeug (z. B. bei Sicherheitsüberprüfungen oder im Forschungsbereich) werden (sog. dual use tools), ist der objektive Tatbestand des Paragraph 202c StGB-E nicht erfüllt. Die bloße Eignung von Software zur Begehung von Computerstraftaten ist daher nicht ausreichend, so dass auch solche Programme aus dem Tatbestand herausfallen, die lediglich zur Begehung von Computerstraftaten missbraucht werden können."

Vorbereitung einer Straftat

Weiterhin heißt es dort: „Zudem muss die Tathandlung zur Vorbereitung einer Computerstraftat (§202a, 202b, 303a, 303b StGB) erfolgen. Entscheidend für die Tatbestandserfüllung des Paragraph 202c StGB-E ist, dass der Täter eine eigene oder fremde Computerstraftat in Aussicht genommen hat. Das ist nicht der Fall, wenn das Computerprogramm beispielsweise zum Zwecke der Sicherheitsüberprüfung, zur Entwicklung von Sicherheitssoftware oder zu Ausbildungszwecken in der IT-Sicherheitsbranche hergestellt, erworben oder einem anderen überlassen wurde, da die Sicherheitsüberprüfung, die Entwicklung von Sicherheitssoftware oder die Ausbildung im Bereich der IT-Sicherheit keine Computerstraftat darstellen."

Sicherheitsanalysen

Auf den zulässigen Einsatz von Analysetools zum Aufdecken von Sicherheitsproblemen zum Beispiel im Zusammenhang mit der Durchführung von Penetrationstestings wird dort wie folgt hingewiesen: „Auch nach der Neufassung des Tatbestandes ist die Verschaffung des Zugangs zu Daten unter Verletzung von Sicherheitsmaßnahmen nur strafbewehrt, wenn der Täter unbefugt handelt. Nicht strafbar ist daher z. B. das Aufspüren von Sicherheitslücken im EDV-System eines Unternehmens, soweit der „Hacker" vom Inhaber des Unternehmens mit dieser Aufgabe betraut wurde."

Trotz der Vorbehalte und starken Kritik hat der Bundestag am 25. Mai 2007 das Gesetz unverändert verabschiedet. Es bleibt abzuwarten, wie sich das Gesetz auf die Praxis der Sicherheitsexperten wirklich auswirken wird, und ob eine Nachbesserung, wie von vielen Kritikern vehement gefordert, beschlossen wird.

Neues Grundrecht

Für vergleichsweise großes Aufsehen hat das so genannte Online-Durchsuchungsurteil des Bundesverfassungsgerichts vom 27. Februar 2008 (1BvR

370/07; 1 BvR 595/07) gesorgt. In diesem Urteil hat das Gericht ein neues Grundrecht auf *Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme* etabliert. Bei der Urteilsverkündung in Karlsruhe erläuterte der Präsident des Bundesverfassungsgerichts, dass dieses neue Grundrecht zu den anderen Freiheitsrechten wie insbesondere dem Recht auf Schutz des Telekommunikationsgeheimnisses, dem Recht auf Unverletzlichkeit der Wohnung und dem Recht auf informationelle Selbstbestimmung hinzu kommt. Dem Urteil ging eine Verfassungsbeschwerde voraus, die sich gegen die Vorschriften im Verfassungsschutzgesetz von Nordrhein-Westfalen zur Online-Durchsuchung richtete. In der Urteilsbegründung haben die Richter unterstrichen, dass eine heimliche Online-Durchsuchung von Rechnern nur unter strengen Auflagen möglich sei, da nach Maßgabe des Bundesverfassungsgerichts das allgemeine Persönlichkeitsrecht auch ein Grundrecht auf Gewährleistung der Vertraulichkeit und Integrität informationstechnischer Systeme umfasst. Das neue Grundrecht ist subsidiär, das heißt, es tritt gegenüber bestehenden Grundrechten zurück (u.a. dem Recht auf informationelle Selbstbestimmung) und soll zur Anwendung kommen, um Schutzlücken abzudecken. Die Konsequenzen und die Bedeutung dieses Urteils werden unter Juristen sehr stark diskutiert. Es ist noch offen, ob das Gesetz ggf. zivilrechtliche und auch haftungsrechtliche Konsequenzen für den Missbrauch von IT-Systemen nach sich ziehen kann und welche Auswirkungen eine derartige Rechtsprechung dann auf die IT-Sicherheitsqualität von IT-Komponenten und Software haben wird.

Dieser kurze Blick auf die Gesetzeslage ist keineswegs erschöpfend, sondern hatte lediglich zum Ziel, das Bewusstsein zu schärfen, dass neben technischen Randbedingungen bei der Konstruktion und Administration sicherer IT-Systeme stets auch eine Vielzahl rechtlicher Regelungen zu beachten und zu befolgen sind. Mit der zunehmenden Globalisierung der Geschäftsprozesse nehmen auch die grenzüberschreitenden, Internet-basierten Aktivitäten von Unternehmen, aber auch von Einzelpersonen stark zu. Hierfür rechtliche Rahmenbedingungen zu erarbeiten ist Gegenstand von Forschungsaktivitäten an juristischen Fakultäten, die häufig unter dem Namen Cyber Law zusammengefasst werden.

Cyber Law

1.4 Computer Forensik

Forensik

Mit der Zunahme der Schadensfälle, die auf absichtliche Angriffe auf IT-Systeme zurückzuführen sind, hat sich in den letzten Jahren unter dem Begriff Computer-Forensik (engl. *computer forensics*) oder auch digitale Forensik eine spezielle Disziplin entwickelt (vgl. u.a. [72]). Methoden der Computer Forensik beschäftigen sich mit dem Nachweis und der Auf-

klärung von ggf. strafbaren Handlungen durch Auswertung von digitalen Spuren. Für einen gerichtsverwertbaren Nachweis (so er denn wirklich geführt werden kann) ist es notwendig, dass die Analysen nicht auf dem Originaldatenbestand, sondern nur auf Kopien durchgeführt werden (never touch original) und dass die Ergebnisse neutral, überprüfbar und nachvollziehbar präsentierbar sind. Forensische Untersuchungen werden nach Möglichkeit nur auf den erstellten forensischen Kopien (forensic sound imaging) und im read-only-Modus durchgeführt. Die Originalsysteme sollten – falls möglich – sicher verwahrt werden. Dadurch wird eine spätere, unabhängige Überprüfung der Analysen möglich.

Allgemeines Vorgehen

Da bei möglichen gerichtlichen Auseinandersetzungen die Beweisketten von Dritten jederzeit überprüfbar nachvollziehbar sein müssen, sind bei der forensischen Analyse Open Source Tools interessant, da deren Code offen gelegt und von Jedermann nutzbar ist. Im Folgenden benennen wir einige derartige Tools. Diese beispielhafte Benennung hat lediglich das Ziel, einen groben Einblick in die Funktionsweise derartiger Tools zu vermitteln. Eine Zusammenstellung frei verfügbarer Forensik-Tools sowohl für Unix-Derivate als auch für Windows-Systeme findet man unter anderem auf der Seite des RUS-CERT der Universität Stuttgart[13].

Imaging

Einer der wichtigsten Schritte bei der forensischen Analyse der Daten ist die Erstellung eines korrekten Abbildes (image) des Datenmediums. Das einfache *dd*-Programm[14], das Bestandteil von Linux-Distributionen ist, erfüllt bereits alle Anforderungen, die z.B. das Computer Forensic Tool Testing Programm der US-Regierung[15] an Imaging-Tools stellt. Es erstellt eine exakte 1:1 Kopie eines Datenträgers. Mit der Erweiterung *dcfldd*[16] wird gleichzeitig auch ein MD5-Hashwert berechnet, der die Daten authentifiziert. Die *dd*-Werkzeuge sind zwar leistungsstark, aber nicht sehr komfortabel in ihrer Bedienung. Diesen Mangel behebt das *AIR*-Tool (Automated Image & Restore[17].

Read-only

Um mit der erstellten Kopie aussagekräftige Analysen vornehmen zu können, ist es notwendig, die Kopie in einem read-only Modus zu mounten, um zu verhindern, dass die Untersuchungsergebnisse durch versehentliche Veränderungen an den Datenbeständen unbrauchbar werden. Unter Linux steht mit dem *Enhanced Linux Loopback*-Tool[18] ein Programm zur

[13] http://cert.uni-stuttgart.de/forensics/
[14] vgl. http://www.gnu.org/software/coreutils/
[15] vgl. http://www.cftt.nist.gov
[16] vgl. http://sourceforge.net/projects/biatchux
[17] vgl. http://air-imager.sourceforge.net/
[18] vgl. ftp://ftp.hq.nasa.gov/pub/ig/ccd/enhanced_loopback/

Verfügung, das ein read-only Mounten des *dd*-Images ganzer Datenträger ermöglicht.

Analyse

Hat man ein exaktes Abbild erstellt, benötigt man geeignete Analysetools, um den Datenbestand zu untersuchen. Hierzu bietet beispielsweise Unix/Linux einige Werkzeuge und Programme wie *grep, strings, find, file, hexedit* standardmäßig an, die aber für die Analyse großer Datenbestände nicht sehr geeignet sind. Unter dem Namen *Sleuth Kit*[19] werden Unix-basierte Kommandozeilen-Programme zusammengefasst, mit denen Dateisysteme wie NTFS, FAT, FFS, ext2fs sowie ext3fs analysiert und insbesondere nach manipulierten Datenbeständen gesucht werden kann.

Fazit

Fazit

Die Computer Forensik gewinnt angesichts der Zunahme von Straftatdelikten im Bereich der IT zunehmend an Bedeutung. Obwohl bereits eine Vielzahl von Tools zur Verfügung steht, um effektiv elektronische Spuren zu sichern, ist die Thematik nicht unproblematisch. Bei der forensischen Analyse sind natürlich auch datenschutzrechtliche Aspekte zu beachten und die Persönlichkeitsrechte des Einzelnen sind zu wahren. Inwiefern forensische Analysen elektronischer Datenspuren, die ja beliebig manipuliert werden können, wobei auch der Manipulationsvorgang selber auch unkenntlich gemacht werden kann, tatsächlich zu einer lückenlosen Beweiskette zusammengeführt werden können, die einer rechtsstaatlichen Überprüfung standhalten, ist jedoch noch offen. Sicherlich ist es für interne Revisionszwecke hilfreich, Anscheinsbeweise anhand sorgfältiger forensischer Analysen vorlegen zu können. Die Durchführung einer solchen Analyse mit seriös nachvollziehbaren Ergebnissen ist jedoch eine schwierige Aufgabe, die ein tiefes Verständnis der Systeme voraussetzt.

1.5 Sicherheitsrichtlinie

Das Spektrum der Sicherheitseigenschaften, die ein System oder eine einzelne Anwendung erfüllen muss, ist breit und abhängig von der Funktionalität, dem Einsatzgebiet, ggf. der unternehmerischen Geschäftsstrategie sowie seiner Nutzung. Die zu realisierenden Sicherheitseigenschaften definieren die Schutzziele eines Systems; diese werden in den Sicherheitsregeln[20] [63] bzw. in Sicherheitsrichtlinien zusammengefasst.

[19] vgl. http://www.sleuthkit.org/sleuthkit/

[20] Im deutschen Sprachgebrauch auch häufig als Sicherheitspolitik bezeichnet. Da die Begriffe der Richtlinien oder Regelwerke den Sachverhalt am Besten beschreiben, werden wir im Folgenden diese Begriffe verwenden.

Definition 1.14 (Sicherheitsrichtlinie)

Die Sicherheitsrichtlinie (engl. *security policy*) eines Systems oder einer organisatorischen Einheit legt die Menge von technischen und organisatorischen Regeln, Verhaltensrichtlinien, Verantwortlichkeiten und Rollen sowie Maßnahmen fest, um die angestrebten Schutzziele zu erreichen. *Richtlinie*

□

Die Sicherheitsrichtlinie einer organisatorischen Einheit wie zum Beispiel eines Unternehmens, einer Abteilung oder eines Projekts besteht in der Regel aus einer Kombination von systembestimmten und benutzerbestimmbaren Anteilen. Systembestimmte Regeln sind globale Regeln, die von einer dafür zuständigen Einheit (z.B. dem Management eines Unternehmens) festgelegt werden. Wir unterscheiden Zugriffskontroll- und Informationsfluss-Richtlinien. In der Praxis werden hauptsächlich Varianten von Zugriffskontroll-Regeln verwendet, wobei zwischen benutzerbestimmbaren, rollenbasierten und systembestimmten Regeln unterschieden wird.

Die Klasse der benutzerbestimmbaren Richtlinien (engl. *discretionary policy*) erlaubt Benutzern, die Rechtevergabe für Objekte, die sie erzeugt haben, individuell zu kontrollieren. *benutzerbestimmbar*

Rollenbasierte Richtlinien reglementieren den Zugriff auf Objekte durch die Festlegung von Rollen, in denen Benutzer bzw. die in seinem Auftrag tätigen Prozesse und Maschinen in dem System aktiv sein dürfen. Eine Rolle benennt eine Aufgabe, die durchzuführen ist, und beschreibt die Berechtigungen und Verantwortlichkeiten, die mit der Rolle verbunden sind. Das bedeutet, dass Zugriffsrechte nicht an individuelle Benutzer, sondern an Rollen vergeben werden und dass Benutzer von einem Systemadministrator als Mitglieder von Rollen einzutragen sind. Die Rollen sollten die vom Benutzer durchzuführenden Arbeiten charakterisieren. *rollenbasiert*

Für die Klasse der systembestimmten, d.h. mandatorischen Richtlinien (engl. *mandatory policy*) gilt, dass die Vergabe von Zugriffsrechten systemglobal auf der Basis von a priori Festlegungen kontrolliert wird. Ein Beispiel für eine entsprechende a priori Festlegung ist die Klassifikation der in einem System zu verarbeitenden Informationen nach Sicherheitsklassen (z.B. öffentlich, geheim, streng geheim) und die Einteilung von Objekten und Subjekten in diese Sicherheitsklassen. Die systembestimmte Richtlinie legt für jedes klassifizierte Subjekt fest, welche Informationsklassen diesem Subjekt maximal zugänglich sein können, unabhängig von den durch das Subjekt tatsächlich durchgeführten Aktionen. Richtlinien, die den Datenzugriff auf der Basis klassifizierter Subjekte und Objekte beschränken, nennt man auch Multi-Level-Sicherheitsrichtlinien. Ein weiteres Beispiel für mandatorische Regeln ist die Festlegung einer unternehmensweit geltenden Richtlinie, die *systembestimmt* *multi-level*

bestimmt, dass E-Mails von Mitarbeitern nur verschlüsselt ausgetauscht werden dürfen und alle Mails auf einem zentralen E-Mail-Server zunächst entschlüsselt und auf Viren überprüft werden, bevor sie an die Adressaten weitergeleitet werden.

informelle Richtlinie

Eine Sicherheitsrichtlinie legt also die gewünschten Sicherheitseigenschaften eines Systems fest, so dass wir die in Definition 1.2 angegebenen Sicherheitsbegriffe entsprechend interpretieren müssen. Die Sicherheit eines Systems ist keine absolute, sondern immer eine relative Eigenschaft, bezogen auf die spezifizierten Sicherheitsregeln. In der Praxis werden Sicherheitsrichtlinien häufig textuell und damit informell festgelegt, so dass eine der Schwierigkeiten darin besteht, derartige Festlegungen auch kontrollierbar umzusetzen. Ein typisches Beispiel ist eine Richtlinie, mit der in Unternehmen der Umgang mit Passwörtern geregelt wird. Nachfolgend geben wir zur Illustration einen Auszug aus einer typischen Passwort-Policy an.

Beispiel: Passwort-Richtlinie (Auszug)

- All system-level passwords (e.g., root, enable, NT admin) must be changed on at least a quarterly basis.
- All user-level passwords (e.g., email, web, desktop computer, etc.) must be changed at least every six months. The recommended change interval is every four months.
- User accounts that have system-level privileges granted through group memberships or programs such as *sudo* must have a unique password from all other accounts held by that user.
- Passwords must not be inserted into email messages or other forms of electronic communication.

Die Thematik des methodischen Erstellens von Regelwerken, des Policy-Engineerings, und systematischen Umsetzens (engl. *enforcement*) von Sicherheitsregeln ist noch Gegenstand aktiver Forschung (u.a. [178]). Wesentliche Fragestellungen hier betreffen die werkzeugunterstützte Erstellung von Regelwerken, zum Beispiel unter Nutzung von Best-Practice Poliy-Patterns (u.a. [179]), die automatische Generierung von maschinell ausführbaren Regelwerken, zum Beispiel als XACML-Code (u.a. [162]), oder aber auch die Entwicklung von Systemarchitekturen zur kontrollierbaren, verteilten Umsetzung von Sicherheitsregelwerken (u.a. [180]).

Mehrseitige Sicherheit

In heutigen vernetzten IT-Systemen kommunizieren eine Vielzahl von Partnern miteinander, die ganz unterschiedliche Schutzziele verfolgen können.

Betrachten wir zum Beispiel einen Content-Provider, der Web-Dienste anbietet. Der Provider ist sicherlich daran interessiert, möglichst viele und präzise Informationen über seine Kunden zu sammeln, um damit Kundenprofile für zugeschnittene Angebote zu erstellen. Der Kunde hat demgegenüber ein berechtigtes Interesse an Privatheit, also an der Preisgabe von möglichst wenigen personenbezogenen Informationen. Dieses Schutzbedürfnis von Benutzern wird in Deutschland auch durch das informationelle Selbstbestimmungsrecht unterstützt, das das Sammeln, Speichern und Verarbeiten personenbezogener Daten durch Dritte gesetzlich regelt (vgl. BVerfGE 65, 1, 49).

Mehrseitige Sicherheit

Der Interessenskonflikt zwischen dem Provider und seinen Kunden ist typisch für Szenarien, in denen mehrseitige (siehe u.a. [145]), einander durchaus auch widersprechende Sicherheitsinteressen zu berücksichtigen sind. Häufig stehen sich hierbei die Sicherheitsinteressen von Herstellern, Betreibern und Nutzern gegenüber. Mit der Zunahme von Telekooperationsanwendungen als Formen der entfernten Zusammenarbeit steigt der Bedarf nach Systemen und Anwendungen, die eine mehrseitige Sicherheit unterstützen. Es werden Systeme benötigt, die es erlauben, dass Kommunikations- und Geschäftspartner ihre beidseitigen gemeinsamen Sicherheitsinteressen aushandeln und sich auf ein gemeinsames Sicherheitsniveau einigen können. Mehrseitige Sicherheit und Wahrung der Privatsphäre spielt insbesondere bei E-Commerce-Transaktionen eine wichtige Rolle. So wurde im Rahmen des BMBF-geförderten Projektes FlexPay ein datenschutzfreundliches Kundenbindungssystem entwickelt (siehe [70]) und mit einem ebenfalls konzipierten, fairen DRM-System gekoppelt (siehe [204]). Das entwickelte DRM-System bringt die Bedürfnisse von Nutzern, die lizensierte digitale Inhalte zur Nutzung an andere weitergeben möchten, mit den Bedürfnissen der Content-Provider, die eine widerrechtliche Content-Nutzung verhindern möchten, in Einklang.

Abbildung 1.4 stellt die eingeführten Begriffe noch einmal miteinander in einen Zusammenhang und zeigt deren Abhängigkeiten auf.

1.6 Sicherheitsinfrastruktur

Die festgelegte Sicherheitsrichtlinie ist unter Nutzung von Sicherheitskonzepten, -mechanismen und -diensten einer Sicherheitsinfrastruktur zu realisieren und zu gewährleisten.

Definition 1.15 (Sicherheitsinfrastruktur)

Architektur

Unter einer Sicherheitsinfrastruktur bzw. einer Sicherheitsarchitektur verstehen wir den Bestandteil einer System-Architektur, der die festgelegten

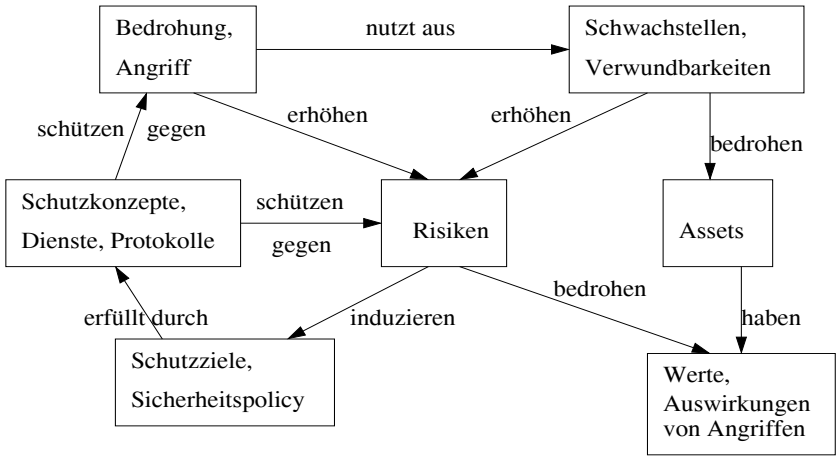

Abbildung 1.4: Zusammenhänge und Abhängigkeiten

Sicherheitseigenschaften durchsetzt und die für die Verwaltung der sicherheitsrelevanten Informationen und Konzepte erforderlichen Realisierungsmaßnahmen zur Verfügung stellt.

□

Abbildung 1.5 gibt einen stark vergröberten Überblick über wesentliche Komponenten (grau unterlegt) einer Sicherheitsarchitektur.

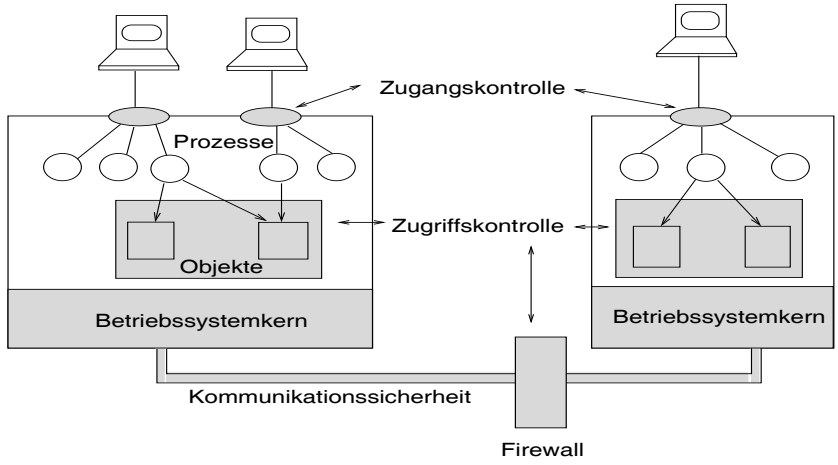

Abbildung 1.5: Komponenten einer Sicherheitsarchitektur (stark vergröbert)

Zugangskontrolle

Der kontrollierte Zugang eines Subjekts zum System erfolgt über die Zugangskontrolle, durch die das Subjekt identifiziert und authentifiziert wird. Auf Mechanismen und Verfahren zur Gewährleistung der Authentizität von Subjekten wird ausführlich in Kapitel 10 eingegangen.

Zugriffskontrolle

Die Zugriffskontrolle hat sicherzustellen, dass ein Zugriff auf ein zu schützendes Objekt nur den dazu berechtigten Subjekten gestattet wird. Der Zugriffskontrolle obliegt sowohl die Gewährleistung der gewünschten Integritätseigenschaften als auch die Durchführung von Informationsflusskontrollen, falls durch die Sicherheitsrichtlinie Vertraulichkeitseigenschaften gefordert sind. Der Betriebssystemkern stellt zusammen mit der Prozessor-Hardware Basismechanismen für diese Kontrollen bereit. Eine spezielle Ausprägung solcher Zugriffskontrollen realisieren Firewall-Rechner. Kapitel 12 widmet sich ausführlich den Techniken zur Durchsetzung der Zugriffskontrolle, während auf die Firewall-Technologie in Kapitel 13.1 eingegangen wird.

Kommunikation

Die Kommunikation in einem vernetzten IT-System erfolgt über Rechnernetze unter Nutzung von Kommunikationsprotokollen. Wesentliche Basismechanismen zur sicheren Kommunikation werden ausführlich in den Kapiteln 7 und 8 behandelt. Deren Einsatz zur Realisierung sicherer Kommunikationsprotokolle ist Gegenstand von Kapitel 13.

Ganzheitliche Sicherheit

Der Bereich der organisatorischen, baulichen und administrativen Maßnahmen zum Schutz eines Systems ist in Abbildung 1.5 nur implizit enthalten. Hierzu gehört der Diebstahlschutz ebenso wie speziell abgeschirmte Räume für den Systemzugang oder die Isolierung von Systemkomponenten in besonders geschützten Gebäudeteilen. Auf die organisatorischen und baulichen Maßnahmen gehen wir im Folgenden nicht ein, da das vorliegende Buch den Schwerpunkt auf die Behandlung der technischen Aspekte der Systemsicherheit legt. Wichtige administrative Aufgaben sowie organisatorische Regelungen, die das technische System selbst betreffen, werden an den entsprechenden Stellen aufgegriffen. Wobei auch hierfür gilt, dass die erschöpfende Behandlung aller Fragen eines ganzheitlichen Sicherheitsmanagements den Rahmen des Buches bei Weitem sprengen würde.

Zu organisatorischen Maßnahmen gehört beispielsweise auch die Etablierung eines Notfallkonzeptes mit einer Planung sowie mit Vorgehensmodellen, wie in der jeweiligen Organisation systematisch und methodisch auf Sicherheitsvorfälle reagiert werden muss. Dies wird häufig unter dem Begriff Incident Handling zusammengefasst. Ferner gehören zu diesem Bereich auch Konzepte und Maßnahmen, um beim Eintreten eines Schadensfalls dafür zu sorgen, dass wichtige, die Interessen der Organisation betreffende Geschäftsprozesse möglichst aufrecht erhalten bleiben bzw. möglichst schnell wieder durchführbar sind. Diese Problematik wird un-

Business Continuity

ter dem Stichwort Business Continuity Planning zusammengefasst. Um derartige Planungen und Strategien in Unternehmen zu etablieren, sind vertiefte Kenntnisse der Geschäftsprozesse des Unternehmens sowie betriebswirtschaftliche Analysen über Kosten und Risiken erforderlich. Auch die Behandlung dieser Thematik, obwohl sie ohne Zweifel sehr wichtig und interessant ist, geht über den Rahmen des vorliegenden Buches hinaus. Aussagekräftige Hinweise für den Bereich der organisatorischen Maßnahmen liefert unter anderem das IT-Grundschutzhandbuch des BSI[21] [37].

Sicherheitskette

Sicherheitskette

Die folgenden Kapitel beschäftigen sich ausführlich mit den wichtigsten Teilbereichen der IT-Sicherheit. Es werden die heute gängigen Maßnahmen, Konzepte und Protokolle beschrieben, die man zur Lösung der jeweiligen Probleme einsetzen kann. Es muss jedoch stets klar sein, dass ein Absichern eines Teilbereichs eines Systems unzureichend ist, um tatsächliche IT-Sicherheit zu gewährleisten. Das Bild der Sicherheitskette verdeutlicht dies sehr gut, da eine Kette nur so stark ist, wie ihr schwächstes Glied.

Abbildung 1.6 veranschaulicht diese Vorstellung. Sie zeigt sicherheitsrelevante Komponenten eines IT-Systems, sowie exemplarische Bedrohungen, die darauf abzielen, Schwachstellen der Komponenten auszunutzen und die Sicherheitskette zu brechen.

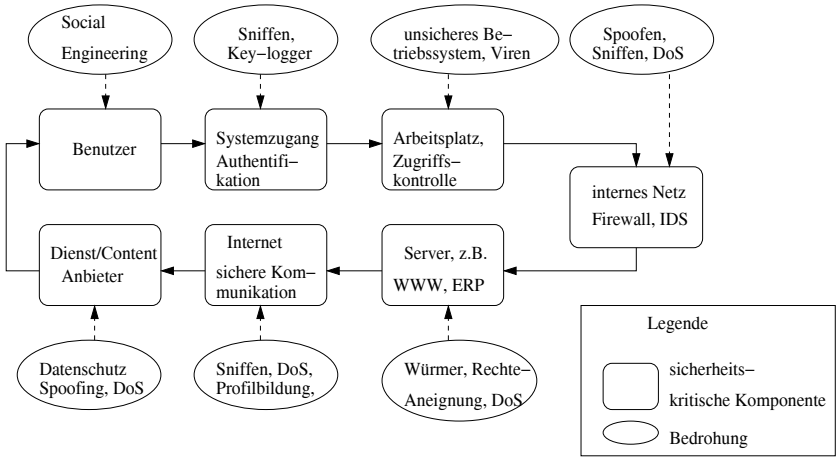

Abbildung 1.6: Sicherheitskette und ausgewählte Bedrohungen

Natürlich kann man diese Kette auch verkürzen, falls in dem real existierenden IT-System Komponenten fehlen (z.B. der Zugriff auf externe Content-Anbieter). Das Bild dient dazu, auf eine sehr vergröberte Weise die

[21] Bundesamt für Sicherheit in der Informationstechnik.

Abhängigkeiten zu veranschaulichen, die zwischen den Systemkomponenten bestehen.

Kettenglieder

Die Kette fängt beim Benutzer an, der natürlich in vielerlei Hinsicht eine Schwachstelle darstellt, was durch die möglichen Social Engineering-Angriffe angedeutet wird. Bevor ein Benutzer Dienste nutzen kann, muss er sich bei einem Rechner bzw. dem System anmelden, also authentifizieren. Hier ergeben sich weitere Angriffspunkte, falls der Authentifizierungsvorgang ungenügend abgesichert ist, sei es, dass zum Beispiel Passwörter im Klartext über das Netz übertragen werden, oder dass durch Key-logger Programme die Tastatureingaben abgefangen werden. Hat sich der Benutzer auf einem Arbeitsplatzrechner (u.a. PC, Laptop, PDA, Workstation) angemeldet, so muss dieser Zugriffskontrollen durchführen, um sicher zu stellen, dass nur autorisierte Daten- bzw. Informationszugriffe möglich sind. Mögliche Bedrohungen der Datenintegrität und Vertraulichkeit ergeben sich u.a. durch Viren oder auch durch unsichere Betriebssystemdienste wie beispielsweise manipulierte Gerätetreiber. Dies setzt sich fort, wenn innerhalb eines Unternehmensnetzes auf Daten zugegriffen werden soll, die sich auf unternehmenslokalen Servern (z.B. Mail-Server, WWW-Server, Enterprise Resource Planning (ERP) Programme) befinden. Dazu wird ein lokales Netz zur Datenkommunikation verwendet, das Bedrohungen u.a. durch Lauschangriffe (Sniffen) ausgesetzt ist. Handelt es sich bei dem lokalen Netz um ein drahtloses Zugangsnetz (WLAN), so sind die Bedrohungen, die durch das Abhören der Funkverbindungen auftreten, offensichtlich. Ist auch von außen ein Zugang über das lokale Netz möglich, so sollten die Daten über geeignete Firewall-Architekturen oder mittels Intrusion Detection Systeme kontrolliert und gefiltert werden. Soll darüber hinaus über ein öffentliches Netz wie dem Internet auf externe Content-Anbieter (u.a. amazon.com, e-commerce Portale) zugegriffen werden, so ist zum einen die Kommunikation dorthin abzusichern und zum anderen muss man darauf vertrauen können, dass die Daten auch auf den externen Servern sicher und u.a. datenschutzkonform verwaltet werden. Beispiele für Bedrohungen sind das Erstellen von datenschutzgefährdenden Benutzerprofilen.

Fordert man ein angemessenes Sicherheitsniveau für ein System, so erfordert das eine ganzheitliche Absicherung der Systeme; eine Fokussierung auf Einzelaspekte (u.a. Beschränkung der Maßnahmen zur IT-Sicherheit auf die Konfigurierung einer Firewall und eines Virenscanners), wie es heute noch immer an der Tagesordnung ist, erweist sich in der Praxis immer wieder als unzureichend. Die Sicherheitskette verdeutlicht, dass z.B. eine sichere Kommunikation nicht ausreicht, um Daten während des gesamten Verarbeitungsprozesses zu schützen. Erlangt ein Benutzer unautorisiert Zugriff auf diese Daten, wenn sie auf einem der beteiligten Rechner verarbeitet oder

gespeichert werden, so bleibt der alleinige Schutz der Kommunikationsverbindung wirkungslos; die getroffenen Sicherheitsmaßnahmen sind nicht ausreichend.

Neue Herausforderungen

Eingebettete Systeme

Das Bild der Sicherheitskette veranschaulicht jedoch auch die Problematik, die sich angesichts der zunehmenden Durchdringung unseres beruflichen und privaten Alltags mit IT-Technologie ergibt. Diese Entwicklung ist auch unter den Begriffen des Ubiquitous Computing, Pervasive Computing oder Ambient Intelligence bekannt. Charakteristisch ist die Einbettung von Sensoren und Aktoren in Alltagsdinge und deren drahtlose Vernetzung. Eine bekannte Ausprägung dieser Entwicklung ist die weite Verbreitung von RFID-Chips, die zur Identifikation von Objekten und zum Überwachen von Objekten (u.a. Container, Lebensmittel, Medikamente, Chemikalien) insbesondere in Logistik-Prozessen verwendet werden. Bereits heute werden 99% aller neu produzierten Halbleiter in eingebetteten Systemen eingesetzt. Nach Schätzung von Roland-Berger[22] liegt in einer modernen Industriegesellschaft der Anteil von Produkten, die eingebettete Systeme enthalten, bei ca. 80% der gesamten Wertschöpfung der verarbeitenden Industrie. Hand in Hand mit der Zunahme eingebetteter Hardware-Komponenten wächst auch das Volumen der darauf eingesetzten Software drastisch. Ein Beispiel mag diese Entwicklung verdeutlichen: Die in einem einzelnen Pkw eingesetzten Softwarekomponenten umfassen heute ca. 10 Millionen Codezeilen, für das Jahr 2015 werden bereits 100 Millionen Lines of Code prognostiziert. Zum Vergleich: Das Betriebssystem Windows XP umfasst lediglich ca. 40 Millionen Codezeilen.

Eingebettete Systeme verfügen häufig über sehr beschränkte Speicher, Rechen- und Energieressourcen, so dass klassische Sicherheitstechniken, wie sie im vorliegenden Buch behandelt werden, in der Regel nicht direkt übertragbar sind. Auch aufwändige Protokolle, zum Beispiel zum Austausch von Kommunikationsschlüssel, können nicht übernommen werden, so dass neue Verfahren für ein effizientes und vor allem auch skalierendes Schlüsselmanagement zu entwickeln sind. Eingebettete Systeme können oder dürfen häufig nach ihrem Ausrollen nicht mehr verändert werden, da sie durchgehend und häufig unter Realzeit-Anforderungen ihre Funktionen (z.B. Überwachungsfunktionen) erfüllen müssen. Die Entwicklung sicherer Change- und Update-Mechanismen ohne dass es dadurch zu einer sicherheitskritischen Beeinträchtigung des Systems und der umgebenden Infrastruktur kommt, sind ebenfalls noch weitestgehend offene Fragestellungen. Teilweise lassen sich existierende Sicherheitslösungen adaptieren,

[22] Vgl. http://www.bitkom.org/files/documents/Zukunft_digitale_Wirtschaft_BITKOM-Roland_Berger_Studie.pdf

teilweise müssen auch existierende IT-Sicherheitsinfrastrukturen verändert werden, um den zusätzlichen Gefährdungen, die sich aus den ubiquitären Rechnern ergeben, zu begegnen.

CPS

Mit dem Vordringen der IKT Technologien in alle Lebens- und Arbeitsbereiche ist eine zunehmende Konvergenz von IT-basierten und physischen Systemen verbunden. Es entstehen so genannte Cyber Physical Systems [221], die häufig in sicherheitskritischen Anwendungen wie im Verkehr und in Transportsystemen, dem Gesundheitswesen, der Energieversorgung oder auch der Logistik eingesetzt werden. CPS sind charakterisiert durch eine große Heterogenität, starke Vernetzung, und eine zunehmende Integration von steuernden Systemen (eingebettete Systeme) in physikalische Umgebungen und betriebliche Prozesse (Physical System). Diese Systeme ermöglichen eine verstärkte Automatisierung und eine Steuerung ganzer Wertschöpfungsnetze in nahezu Echtzeit. CPS eröffnen ein sehr großes Potential an technologischen Innovationen, neuen Anwendungen und Geschäftsmodellen, wie in dem acatech-Roadmap Projekt *agendaCPS*[23] ausführlich dargestellt. Die Voraussetzung dafür ist, dass die Daten rechtzeitig, vollständig, nicht manipuliert und ggf. auch vertraulich an die berechtigten Empfänger gelangen. Die Frage der Sicherheit ist deshalb für Cyber-Physical Systems von sehr großer Bedeutung. Heterogene Komponenten und heterogene Vernetzungstechnologien werden im Internet der Zukunft unter dem gemeinsamen Dach des IP Protokolls zusammengeführt (All-IP) und sind damit den bekannten Gefährdungen ausgesetzt. Derartige Systeme müssen also hohe Sicherheits- und Zuverlässigkeitsanforderungen erfüllen. Aktuelle Angriffe wie der Stuxnet-Wurm im Jahr 2010 [129] haben jedoch verdeutlicht, dass Cyber-Physical Systeme heute noch in hohem Maße verwundbar sind. Offene Forschungsfragen betreffen beispielsweise die Entwicklung und Anpassung vertrauenswürdiger Hardware-naher, eingebetteter Sicherheitslösungen, die als robuste und angriffsresistente Bausteine in physische Komponenten integriert werden können. Die Entwicklung von neuen Konzepten, Protokollen und Lösungen, zur Absicherung vernetzter Cyber-Physical Systeme und die frühzeitige Erkennung von möglichen Störungen und Angriffen stellen weitere zentrale Forschungsfragen von hoher wirtschaftlicher Relevanz dar. Angriffsversuchen müssen mit hoher Treffergenauigkeit frühzeitig erkannt werden, um eine kaskadierende Ausbreitung mit den entsprechenden Schäden zu verhindern.

Cyber Security

Unter dem Begriff Cyber-Sicherheit oder auch Cyber Security fasst man die Herausforderungen in Bezug auf Sicherheit zusammen, die sich aus der Vernetzung von IKT Systemen und den zunehmenden Abhängigkeiten von vernetzten, sicherheitskritischen Infrastrukturen ergeben. Die Gesamt-

[23] vgl. www.acatech.de/cps

heit der IT-Infrastrukturen in Unternehmen, Behörden, oder aber auch in Produktionsanlagen, im Gesundheitswesen oder im Finanzbereich und der Logistik, die über das Internet oder vergleichbare Vernetzungstechnologien zugreifbar sind, werden als Cyber-Raum bezeichnet. Unter Cyber-Sicherheit versteht man deshalb auch die Ausweitung der Aufgaben der klassischen IT-Sicherheit auf den ganzen Cyber-Raum. In der öffentlichen Diskussion wird derzeit im Kontext Cyber-Sicherheit noch vordringlich die Frage diskutiert, mit welchen organisatorischen, technischen und rechtlichen Maßnahmen die Kriminalität, Wirtschaftsspionage und Sabotage aus dem Netz eingedämmt und eine höhere Transparenz in Bezug auf Sicherheitsvorfälle erzielt werden kann. Im Vordergrund der auf der Bundesebene geführten Debatte steht dabei die Verbesserung des Lagebildes über die Gefährdungslage in Deutschland, wofür im Jahr 2012 die Allianz für Cyber-Sicherheit[24] als eine gemeinsame Initiative des BSI und des BITKOM[25] gestartet wurde. Eine verbesserte Informationslage, um mögliche Gefährdungen frühzeitig zu erkennen und Abwehrmaßnahmen ergreifen zu können, ist sicherlich sehr wichtig. Langfristig tragfähige Lösungen erfordern jedoch die Entwicklung neuer Sicherheits-Technologien, die den Anforderungen hochgradig, u.U. spontan vernetzter und eingebetteter Systeme, z.B. in Bezug auf Energie-Effizienz oder Realzeit-Anforderungen, erfüllen. Neue Sicherheitskontroll- und Schutzmaßnahmen müssen bereits frühzeitig in den Entwurf der Systeme integrierte werden, um zukünftige IT-basierte Produkte und Systeme robuster und resistenter gegen insbesondere auch Internet-basierte Angriffe zu gestalten.

Zur Lösung aller dieser offenen Fragen sind Kenntnisse über die klassischen Probleme in der IT-Sicherheit sowie der Konzepte, Protokolle und Verfahren zu deren Lösung, wie sie in den nachfolgenden Kapiteln erläutert werden, unumgänglich.

[24] http://www.allianz-fuer-cybersicherheit.de
[25] Bundesverband der Informationswirtschaft, Telekommunikation und neue Medien e.V.

2 Spezielle Bedrohungen

Kapitelüberblick

Das Kapitel widmet sich Klassen von Bedrohungen, die Ursache für die in heutigen Systemen am häufigsten auftretenden Sicherheitsprobleme sind. Dazu zählen Bedrohungen durch Pufferüberlauf-Angriffe, Viren, Würmer und Trojanische Pferde sowie mobile Applikationen (Apps). Nicht abgefangene Pufferüberläufe (engl. *Buffer Overflow*) sind sehr weit verbreitete Schwachstellen, die durch nachlässige Programmierung sowohl in Betriebssystemen als auch in Anwendungsdiensten auftreten und häufig durch Würmer ausgenutzt werden. Abschnitt 2.2 erläutert die Buffer-Overflow-Problematik, erklärt die prinzipielle Funktionsweise von Exploits, die solche Schwachstellen ausnutzen, und diskutiert Abwehrmaßnahmen. Abschnitt 2.3 beschäftigt sich dann mit dem allgemeinen Aufbau und der Funktionsweise von Computerviren und gibt Hinweise auf wirksame Maßnahmen zur Virenabwehr. Die charakteristischen Eigenschaften von Würmern zusammen mit möglichen Abwehrmaßnahmen werden in Abschnitt 2.4 behandelt. Abschnitt 2.5 widmet sich in gleicher Weise der Thematik der Trojanischen Pferde. In den letzten Jahren wurden zunehmend Angriffe über sogenannte Bot-Netze durchgeführt. Deren Bedeutung und Gefährdungspotential wird auch in den kommenden Jahren weiter zunehmen. Abschnitt 2.6 erläutert die prinzipielle Arbeitsweise eines Bot-Netzes. Abschließend gehen wir in Abschnitt 2.7 kurz auf die Problematik des mobilen Codes ein. Als mobilen Code oder aktive Inhalte (engl. *active content*) bezeichnet man allgemein alle Programme, die zwischen Rechnern ausgetauscht und in Gastumgebungen ausgeführt werden. Mobile Applikationen, Apps, die in hoher Zahl auf Android-Smartphones oder iPhones verwendet werden, sind neueste Ausprägungen dieses Trends zum mobilen Code.

2.1 Einführung

Schadsoftware

Spezielle Klassen von Bedrohungen eines IT-Systems gehen von Viren, Würmern und Trojanischen Pferden (u.a. [62]) aus, die wir als Schadsoftware (engl. *malware* oder *malicious code*) bezeichnen. Es handelt sich hierbei um Programme, die festgelegte, dem Benutzer jedoch verborgene Funktionen ausführen. Die Festlegung der zusätzlichen und beabsichtigten (i.d.R.

böswilligen) Funktionalität unterscheidet diese Klasse von Programmen von den so genannten Wanzen (bugs), deren Fehlverhalten meist von nicht beabsichtigten Programmierfehlern verursacht wird. Der Befall von IT-Systemen durch Viren, Würmer und Trojanische Pferde führt jedes Jahr zu erheblichen wirtschaftlichen Schäden. So hat das amerikanische Computer Security Institute ermittelt, dass 2001 allein in den USA Schäden durch Viren und Würmer in Höhe von über 50 Millionen Dollar aufgetreten sind. Die Tendenz ist nach wie vor steigend. So schätzt die US-Beratungsfirma Computer Economics[1] den Schaden, den Schadsoftware 2006 weltweit angerichtet hat, auf über 13 Milliarden Dollar. Im Jahr 2008 hat die Firma Symantec 1 656 227 neue Auftreten von Schadsoftware erkannt. Das bedeutet einen Anstieg um 265 Prozent im Vergleich zu 2007[2].

Architektur-Eigenschaften

Basiskonzepte und -eigenschaften heutiger Rechnerarchitekturen erleichtern das erfolgreiche Durchführen von Buffer-Overflow-Angriffen bzw. ermöglichen die Existenz von Trojanischen Pferden und die Verbreitung von Viren und Würmern. Zu nennen ist dabei die gemeinschaftliche (aber nicht notwendigerweise gleichzeitige) Nutzung von Programmen und Daten (engl. *sharing*). Eine solche gemeinsame Nutzung wird u.a. über eine zum Teil sehr freizügige Netzwerkfreigabe oder über öffentliche Verzeichnisse ermöglicht. Die gemeinsame Nutzung von Ressourcen ist ebenfalls ein Charakteristikum von Peer-to-Peer Overlay-Netzen, die sich seit einiger Zeit etablieren. Weit verbreitet sind Musiktauschbörsen und File-Sharing Systeme wie Gnutella, KaZaA oder BitTorrent. Die Eigenschaft der gemeinsamen Ressourcennutzung wird insbesondere von Viren zur Verbreitung ausgenutzt. Eine weitere wesentliche Eigenschaft ist die universelle Interpretierbarkeit von Daten. Sie besagt, dass kein Unterschied zwischen Daten und Programmen besteht. Programme können demnach wie Daten leicht modifiziert werden, indem man sie beispielsweise um einen Schadensteil erweitert. Die universelle Interpretierbarkeit bedeutet aber auch, dass in Speicherbereiche, die eigentlich ausschließlich Daten beinhalten wie beispielsweise die Laufzeitkeller (engl. *stack*) von Prozessen, gezielt ausführbarer Code geladen werden kann. Werden diese binären Daten nun als ausführbarer Code interpretiert, so kann auf diese Weise Schadsoftware zur Ausführung gelangen. Dies wird von den Buffer-Overflow-Angriffen ausgenutzt.

Sharing

universelle Interpretation

Als dritte wesentliche Eigenschaft ist die Transitivität des Informationsflusses zu nennen. Falls eine Information vom Rechner A zum Rechner B fließt und es Möglichkeiten gibt, Informationen von B zu einem Rechner C fließen

Transitivität

[1] Quelle: Studie von Computer Economics http://www.welt.de/die-welt/article1037708/Computer_werden_zu_Zombies.html

[2] Quelle: Symantec Threat Report, http://www.symantec.com/threatreport/

zu lassen, so kann die Information auch von A zu C weitergeleitet werden. Die Transitivität wird insbesondere von Würmern ausgenutzt, um sich auf diese Weise zu verbreiten.

Techniken zur Bedrohungsabwehr

Zur Abwehr der Bedrohungen kann man drei Ansätze verfolgen, die in der Praxis in der Regel in verschiedenen Kombinationen eingesetzt werden. Zu nennen sind an erster Stelle Techniken zur Verhinderung (engl. *prevention*) von Angriffen wie beispielsweise die Verwendung stark typisierter Programmiersprachen, Code-Verifikation oder die Beschränkung von Zugriffen aufgrund von systembestimmten Regeln (z.B. kein Schreiben mit anschließendem Ausführen von Daten auf dem Stack-Segment). Im weitesten Sinn gehören in diese Klasse von Techniken auch Maßnahmen zur Sensibilisierung und Bewusstseinsbildung insbesondere auch von Entwicklern von Software-Systemen. Klar ist, dass Vermeidungstechniken nach dem Motto „Sicherheit per Design“die konzeptuell befriedigenste Lösung darstellen, wir aber heute in der Praxis der System-Konstruktion hiervon noch sehr weit entfernt sind.

Verhinderung

Die zweite Klasse von Ansätzen beschäftigt sich deshalb mit Techniken zur Erkennung und frühzeitigen Abwehr möglicher Bedrohungen. Als Beispiele seien statische Analysen des Programmcodes genannt, um z.B. Code-Stücke, die eine Buffer-Overflow-Schwachstelle aufweisen (u.a. Kopieren von Zeichenfolgen ohne die Längenangaben zu prüfen), zu identifizieren und zu korrigieren. Aber auch die bekannten Viren-Scanner und Intrusion Detection Systeme gehören in diese Klasse von Techniken, die anhand charakteristischer Merkmale versuchen, potentielle Schadsoftware zu erkennen.

Erkennung und Behebung

Zu der dritten Klasse von Ansätzen zählen Techniken, die darauf abzielen, die Auswirkungen von eingetretenen Bedrohungen zu begrenzen (engl. *mitigation*). Dazu zählen Isolierungsansätze wie das Sandboxing, die darauf abzielen, Anwendungsbereiche gegeneinander abzuschotten, so dass ein Schaden, der bei der Ausführung eines Anwendungsprogramms auftritt, sich nicht auf andere Anwendungsbereiche ausbreiten und womöglich das ganze System beeinträchtigen kann.

Schadensbegrenzung

2.2 Buffer-Overflow

Buffer-Overflow-Angriffe nutzen Schwachstellen aus, die sich aus Implementierungsfehlern als Folge einer nachlässigen Programmierung ergeben. Der Ansatzpunkt für entsprechende Angriffe sind Programme, in denen Daten in einen Bereich einer Variablen fester Länge, z.B. in einen String oder ein Feld fester Länge, eingelesen bzw. kopiert werden, ohne dass das Programm prüft, ob die kopierte Eingabe überhaupt in den bereitgestell-

Buffer-Overflow

ten Bereich passt. Derartige Variablen, in die Daten kopiert werden, sind abstrakt gesehen Pufferbereiche, woraus sich der Name dieses Angriffs ableitet. Durch das Schreiben einer zu großen Datenmenge in den Bereich des Puffers wird dieser zum Überlauf (engl. *overflow*) gebracht.

2.2.1 Einführung

Wie auch bei den Viren, so kann man auch bei den Buffer-Overflow-Angriffen verschiedene zeitliche Entwicklungsphasen unterscheiden. Während die erste Generation der Angriffe vordringlich Operationen zum Kopieren von Zeichenketten für Angriff ausnutzten, werden heute zunehmend auch einzelne Integer-Zahlen zum Überlaufen gebracht oder es werden Anweisungen, wie Schleifen, die nicht korrekt terminieren, für Überlauf-Angriffe missbraucht.

C, C++

Schwachstellen, die für Buffer-Overflow Angriffe ausgenutzt werden können, finden sich häufig in Programmen, die in den Programmiersprachen C bzw. C++ geschrieben sind. Probleme ergeben sich beispielsweise durch die Verwendung von Funktionen zur Verarbeitung von Zeichenketten, wie `strcpy()`. Diese Funktion kopiert Zeichen für Zeichen des Eingabestrings auf den Zielstring, ohne dabei zu prüfen, ob der Bereich des Zielstrings groß genug ist, um die Eingabe zu speichern. Der Kopiervorgang terminiert erst dann, wenn im Eingabestring der ASCII-Wert 0 gelesen wird. Auf diese Weise ist es möglich, gezielt vorhandene Werte von Variablen, die auf dem Speicherbereich abgelegt sind, der sich unmittelbar an den Bereich des Zielstrings anschließt, mit eingeschleusten Werten zu überschreiben.

Häufig sind auch Funktionen wie *sizeof()* Ausgangspunkt für eine Buffer-Overflow Verwundbarkeit. So wird beispielsweise bei der Transformation von Daten in Unicode ein Datenstring einer bestimmten Länge, die durch das Kommando *sizeof()* bestimmt wird, in einen anderen Datenstring kopiert. In der Unicode-Transformation wird die Größenangabe jedoch in Byte und nicht in der Anzahl der Characters berechnet, so dass bei einer falschen Verwendung viel mehr Speicher überschrieben wird als vorgesehen war. Ein weltweit bekanntes Beispiel eines Angriffs, der eine derartige Schwachstelle ausnutzte, war der Code-Red Wurm im Jahre 2001, der einen Pufferüberlauf in Microsofts IIS (Internet Information Service) ausnutzte, um das System zu befallen.

Wie bereits weiter oben kurz angedeutet, werden heutzutage auch zunehmend Schwachstellen ausgenutzt, die sich durch die unsichere Programmierung einer Loop-Schleife ergeben. Häufiges Beispiel ist hier das zeichenweise Kopieren eines Eingabestrings in dieser Loop-Schleife, z.B. das Kopieren einer URL, wobei als Abbruchkriterium der Schleife das Lesen eines bestimmten Zeichens festgelegt wird. Fehlt dieses Abbruchzeichen

in der Zeichenkette oder taucht es gezielt erst nach einer Menge von anderen Daten auf, so werden alle diese Daten Zeichen für Zeichen kopiert. Der bekannte Blaster-Wurm (siehe Seite 71) hat beispielsweise eine solche Schwachstelle ausgenutzt.

Da unter anderem alle UNIX-Derivate (u.a. BSD-UNIX, Solaris, HP-UX, Linux) und die aktuellen Windows-Versionen (u.a. Windows 2000, Windows XP, Vista, Windows 7) in C bzw. C++ geschrieben sind, bilden immer wieder fehlerhaft programmierte Betriebssystemdienste, Serverroutinen oder auch Bibliotheksfunktionen Angriffspunkte für Buffer-Overflows.

Ansatzpunkt

In einem Programmfragment in der Programmiersprache C könnte ein möglicher Ansatzpunkt für Buffer-Overflow Angriffe wie folgt gegeben sein:

```
cmd =lese_aus_netz();
do_something(cmd);
int do_something(char *string){
    char buffer[4]; /* Pufferbereich */
    strcpy(buffer,string);  /* Ansatzpunkt: */
     /* Einlesen von Daten in Pufferbereich */
     .....
    return 0;
}
```

Adressraumverwaltung

virtueller Adressraum

Um die Vorgehensweise und Ziele eines Buffer-Overflow-Angriffs verstehen zu können, muss man sich zunächst einmal klarmachen, auf welche Weise in heutigen Betriebssystemen die Daten eines Prozesses, also eines in Ausführung befindlichen Programms, verwaltet werden. Alle gängigen Betriebssysteme unterstützen heute virtuellen Speicher. Jedem Prozess ist ein virtueller Adressraum zugeordnet, der – wie in Abbildung 2.1 skizziert – vom Betriebs- und Laufzeitsystem verwaltet wird.

Segmente

Typischerweise wird der virtuelle Adressraum in drei logische Bereiche, die Segmente genannt werden, aufgeteilt, wobei das Text- bzw. Codesegment die ausführbaren Befehle des Prozesses enthält. In der Regel wird dieses Segment vom Betriebssystem gegen ein Überschreiben geschützt. Auf dem Halden-Bereich (engl. *heap*) werden zur Programmlaufzeit dynamisch erzeugte Datenobjekte abgelegt, aber auch globale Variablen sowie globale Konstanten. Typische dynamische Objekte sind Zeigerstrukturen oder Objekte in objektorientierten Programmiersprachen. Der Heap-Bereich wächst normalerweise von niedrigen Adressen zu höheren Adressen. In der Regel ist die Trennlinie zwischen dem Heap und dem Stack-Segment fließend.

Heap

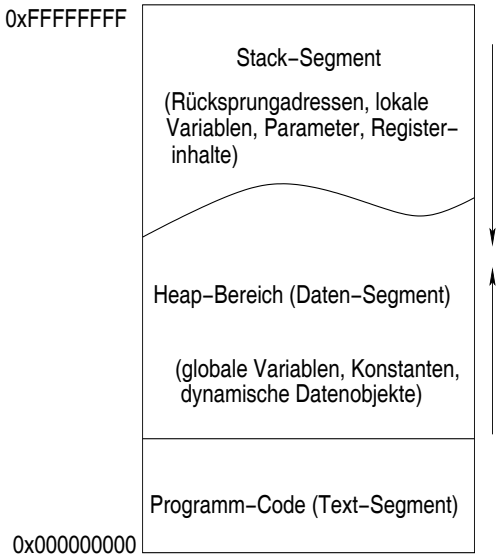

Abbildung 2.1: Segmentierung des virtuellen Prozessadressraums

Stack

Der Laufzeitkeller, das Stacksegment (engl. *stack*), wird LIFO[3]-artig verwaltet und dient zur Verwaltung von Prozeduraufrufen. Auf dem Stack werden u.a. lokale Variablen von Prozeduraufrufen, deren Eingabeparameter sowie die Rücksprungadresse des Aufrufs abgelegt. Der Stack-Bereich wächst von hohen zu niedrigen Adressen; er wächst also dem Heap entgegen. Nach Beendigung eines Prozeduraufrufs werden die lokalen Daten, Parameter und sonstigen Umgebungswerte des Aufrufs wieder vom Stack entfernt. Bei der Programmübersetzung erzeugt der jeweilige Compiler Maschinencode, um diese Daten auf Stack und Heap-Bereichen zu speichern bzw., wie im Falle des Stack-Bereichs, sie auch wieder zu entfernen.

Schutz?

Im Unterschied zum Code-Segment können weder das Stack- noch das Heap-Segment vom Betriebssystem gegen Überschreibungen geschützt werden, da ein schreibender Zugriff ja gerade beim Ablegen von dynamischen lokalen bzw. globalen Daten benötigt wird. Gravierender ist, dass die Daten in diesen Bereichen in der Regel nicht nur gelesen und geschrieben werden dürfen, sondern auch ausführbar sind. Wir werden bei der Diskussion der Gegenmaßnahmen hierauf zurückkommen.

2.2.2 Angriffe

Wie bereits einführend erwähnt, ist es das Ziel eines Buffer-Overflow-Angriffs den Bereich, der für eine Variable im Speicher reserviert ist, zu überschreiben und in der Regel eben auch zum Überlaufen zu bringen. Der-

[3] Last in first out

artige Daten können prinzipiell an unterschiedlichen Stellen im Speicher abgelegt werden. Dazu zählt in erster Linie das Stack-Segment, auf dem lokale Variablen sowie Eingabeparameter, die bei Prozedur- bzw. Methodenaufrufen verwendet werden, abgelegt werden. Es kann sich aber ebenso um den Heap-Bereich des Prozess-Adressraums handeln, auf dem die dynamisch erzeugten Datenobjekte liegen, oder aber auch um einen Bereich handeln, der von mehreren Prozessen gemeinsam genutzt und an jeweils unterschiedliche Stellen des virtuellen Adressraums des einzelnen Prozesses eingeblendet wird. Im Folgenden konzentrieren wir die Beschreibung von Angriffen auf die Stack-Bereichs-Überläufe, da sie sehr häufig auftreten. Aber natürlich muss man sich der anderen Angriffspunkte stets bewusst sein.

Buffer-Overflows auf Stack-Segmenten

überschreiben

Ein Buffer-Overflow-Angriff zielt nun darauf ab, über Eingabeparameter bei einem Prozeduraufruf den Speicherbereich derjenigen lokalen Variable (d.h. den Puffer) auf dem Stack, die diese Eingabe aufnehmen soll, zum Überlauf zu bringen. Das Ziel besteht dann meist darin, durch einen gezielt konstruierten Eingabestring die auf dem Stack abgelegte Rücksprungadresse des Prozeduraufrufs zu überschreiben (vgl. Abbildung 2.2). Hierbei wird wesentlich die Organisation des Stacks ausgenutzt, der wie gesagt von hohen zu niedrigen Adressen wächst. Bei einem Prozeduraufruf werden stets zunächst die Verwaltungsdaten wie z.B. die Rücksprungadresse auf dem Stack abgelegt, bevor Speicherplatz für die lokalen Variablen, die zur Prozedurausführung benötigt werden, angelegt wird. Bei der Speicherung von aktuellen Werten in diesen reservierten Speicherbereichen, also z.B. bei der Übernahme der Eingabewerte eines Eingabstrings, werden die Daten beginnend bei den niedrigen Anfangsadressen der lokalen Variablen auf dem Stack abgelegt.

An Bild 2.2 kann man sich unmittelbar klarmachen, dass durch eine zu lange Eingabe die gespeicherten Werte, die auf dem an den für die lokale Variable reservierten Adressbereich unmittelbar angrenzenden Bereich abgelegt sind, überschrieben werden.

Einschleusen von Fremd-Code

Fremd-Code

Nach dem Überschreiben der Rücksprungadresse kann der Fall vorliegen, dass die überschriebenen Speicherzellen keine sinnvolle Adresse mehr enthalten, so dass das Programm bei der Ausführung des Rücksprungbefehls aufgrund eines Speicherzugriffsfehlers (engl. *segmentation fault*) abstürzen wird. Damit hat der Angreifer zwar u.U. bereits großen Schaden verursacht, aber ggf. noch nicht sein eigentliches Ziel, ausführbaren Code auf dem Stack zu platzieren, erfolgreich erreicht. Der Angreifer wird also versuchen, seinen Eingabestring so zu konstruieren, dass über die Rücksprungadresse die Anfangsadresse des eingeschleusten Schadcodes oder ein anderer, bereits

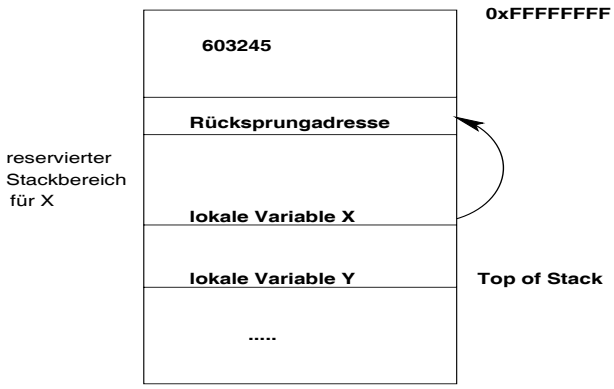

Abbildung 2.2: Buffer-Overflow mit Überschreiben der Rücksprungadresse

geladener Code angesprungen und dann ausgeführt wird. Der eingeschleuste Code wird mit den Berechtigungen des Prozesses ausgeführt, dessen Stack durch den Buffer-Overflow manipuliert wurde. Da der Angreifer auf diese Weise seine Zugriffsrechte vermehrt und im Falle von befallenen Systemprozessen auch anhebt, spricht man auch häufig von Elevation of Privileges-Attacken. Ein erfolgreicher Buffer-Overflow-Angriff ist nur ein Beispiel dafür, wie man als Angreifer seine Rechte ausweiten kann.

Das Problem, das sich hierbei ergibt, besteht darin, dass der Angreifer die genaue, d.h. die absolute Adresse seines eingeschleusten Codes auf dem Stackbereich kennen muss, um gezielt dorthin zu springen. Die absolute Adresse kann jedoch nicht unbedingt vorab ermittelt werden, da sie unter anderem davon abhängig ist, wie viele lokale Variablen und Registerinhalte auf dem Stack abgelegt werden. Für dieses Problem gibt es jedoch eine einfache Lösung. In den zu präparierenden Eingabestring kann der Angreifer eine Anzahl von No-Operation Befehlen (NOP) einstreuen und die Rücksprungadresse mit der Adresse eines dieser NOP-Befehle überschreiben. Da Prozessoren NOP-Befehle einfach ignorieren und überspringen, wird automatisch der nächste Befehl angesprungen bis der eingeschleuste Code erreicht ist (vgl. Abbildung 2.3)

NOPs

komplexe Funktionalität

Das nächste Problem, das ein Angreifer nun haben könnte, ist der relativ begrenzte Speicherbereich, der ihm mit dem Stack als Speicherort für seinen eingeschleusten Code zur Verfügung steht. Dies ist aber nur ein kleines Hindernis, da zum einen mit hoch optimiertem Assemblercode komplexe Programme erstellt werden können, die nur einen geringen Speicherbedarf besitzen. Zum anderen stehen dem Angreifer viele Funktionen direkt zur Verfügung, ohne dass er sie selber einschleusen muss. So liegen in Windows-Systemen viele Bibliotheken (DLLs) bereits im Speicher. Ferner stehen dem Angreifer alle diejenigen Funktionen des Windows API zur Verfügung,

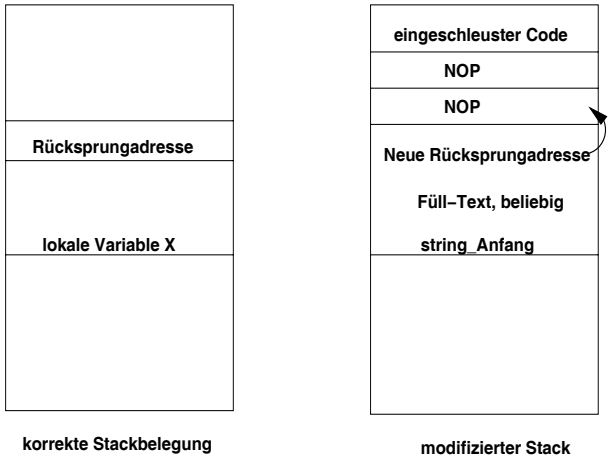

Abbildung 2.3: Einschleusen von Code mit NOPs-Befehlen

die bereits zu dem attackierten Programm hinzu gebunden sind. Befindet sich unter diesen gebundenen Funktionen beispielsweise auch die Funktion `LoadLibrary`, so können durch den Exploit sogar beliebige Funktionen dynamisch nachgeladen werden. Da Bibliotheken häufig an gleichen Stellen im Speicher abgelegt werden, kann ein Angreifer diese direkt aufrufen, um beispielsweise TCP/IP-Verbindungen aufzubauen oder Befehle einer Shell auszuführen.

2.2.3 Gegenmaßnahmen

sichere Programmierung

Aus dem Besprochenen wird klar, wie man sich als Software-Entwickler vor Buffer-Overflow-Angriffen schützen kann. Durch sorgfältige Programmierung (u.a. [212]), bei der insbesondere alle Eingabewerte geprüft und Bereichsgrenzen kontrolliert werden, kann man verhindern, Ziel von gezielten Buffer-Overflow-Angriffen zu werden. Wie bereits eingangs erwähnt, treten besonders in Programmen der Programmiersprachen C und C++ Pufferüberlaufprobleme auf. Durch den Übergang auf Programmiersprachen mit einer höheren Programmiersicherheit wie Java (u.a. [8]), könnten die entsprechenden Probleme überwunden werden. So überprüft beispielsweise das Java-Laufzeitsystem die Einhaltung von Bereichsgrenzen automatisch. Aber auch im C bzw. C++ Umfeld gibt es Möglichkeiten, die Programmiersicherheit zu erhöhen. So kann man beispielsweise anstelle der bereits angesprochenen unsicheren Funktion `strcpy()` die Funktion `strncpy` verwenden, die es ermöglicht, die Zahl der zu schreibenden Zeichen zu begrenzen. Da die Funktion `strcpy()` aber nur eine von vielen derartigen problematischen Bibliotheksfunktionen ist, sollte man eine spezielle Bibliothek verwenden, durch die sichere Bibliotheksfunktionen zur Verfügung gestellt werden. Unter Linux leistet dies beispielsweise die Bibliothek

Wrapper

`Libsafe`[4], mit der Standard-C Bibliotheksaufrufe gekapselt (engl. *to wrap*) und durch die gesicherte Version ersetzt werden.

Compilerunterstützung

Hat man den Quellcode von Programmen zur Verfügung, so kann man diesen unter Zuhilfenahme von speziellen Tools bearbeiten, um Überlaufangriffe zu erkennen. Ein Beispiel hierfür ist das `StackGuard Tool`[5] für UNIX bzw. StackShield für Linux, das im Prinzip einen Patch für den GNU C-Compiler bereitstellt. Der derart modifizierte Compiler sichert bei jedem Funktionsaufruf die Returnadresse und korrigiert sie bei Bedarf. Der Stack-Smashing-Protector, der im GNU C-Compiler seit der Version 4.1 verfügbar ist, fügt ein spezielles Kontrollzeichen, das so genannte Canary[6], direkt hinter die Rücksprungadresse auf den Stack ein. Ferner erzeugt er Code, der vor dem Rücksprung prüft, ob das Kontrollzeichen verändert wurde. Wird eine solche Änderung erkannt, so wird automatisch eine Warnmeldung in die Log-Datei des Systems geschrieben und das Programm abgebrochen. Ein analoges Konzept wurde mit den Stack Cookies von Microsoft u.a. in den Windows Server 2003 und Vista integriert.
Natürlich sind solche Maßnahmen unbefriedigend, da damit ja nicht die Ursache des Problems behoben, sondern nur die Auswirkungen begrenzt werden. Da die Absicherung von Systemen gegen Angriffe von außen ein ständiger Wechsel zwischen Aktionen seitens der Angreifer und Reaktionen seitens der Software-Entwickler und Systemadministratoren bedeutet, ist klar, dass die skizzierten Compilermodifikationen keinen nachhaltigen Effekt haben werden. Angreifer werden ihre Angriffe daraufhin verfeinern und dafür sorgen, dass das eingefügte Kontrollzeichen trotz Manipulation erhalten bleibt.

proprietärer Code

Der gerade beschriebene Ansatz setzt voraus, dass die abzusichernden Programme im Quellcode vorliegen, so dass sie mit den entsprechend modifizierten Compilern erneut übersetzt werden können. Eine Neu-Übersetzung der Software ist aber häufig gar nicht möglich, wie zum Beispiel bei proprietären Betriebssystemen oder bei kommerzieller Anwendungssoftware.

ASLR, DEP

Mit den Techniken ASLR (Address Space Layout Randomization) und DEP (Data Execution Prevention) stehen in herkömmlichen Betriebssystemen Konzepte zur Verfügung, um den Schaden von erfolgreich durchgeführten Buffer-Overflows zu begrenzen bzw. Angriffe prinzipiell zu erschweren. Mit DEP werden Speicherbereiche als nicht-ausführbar gekennzeichnet, so dass kein Code, insbesondere kein Schadcode in einem solchen Datensegment ausgeführt werden kann. Ein entsprechender Versuch, Code auszuführen, löst eine Speicherschutzverletzung aus.

---

[4] http://http://www.research.avayalabs.com/gcm/usa/en-us/initiatives/all/nsr.htm

[5] http://immunix.org

[6] In Anlehnung an das frühere Vorgehen in Bergwerken, wo man einen Kanarienvogel unter Tage mitgenommen hat. Sobald dieser aufhörte zu singen bzw. gar starb war klar, dass ein bedrohlicher Sauerstoffmangel herrschte.

Die ASLR-Technik erschwert Angriffe, die Kenntnisse über das Speicherlayout nutzen, um den Speicher zu manipulieren. Beispiele für solche Angriffe sind neben den beschriebenen Buffer-Overflows auch Heap Overflows bzw. auch Underflows, Format-String Verwundbarkeiten, oder auch Array Index Overflows. Die randomisierte Speicherung von ausführbarem Binärcode, von Dynamic Link Libraries (DLLs), Text-, Heap- und teilweise auch Stack-Segmenten mittels der ASLR-Technik (vgl. u.a. [191]) verschleiert deren Speicheradresse, die für die genannten Klassen von Angriffen oder auch für Angriffe unter Nutzung von ROP (Return-oriented Programming) notwendig ist, um die gespeicherten Module aufrufen zu können. Die ASLR Technik wurde als erstes im Betriebssystem OpenBSD verwendet. Sie wird seit Windows Vista auch mit Windows-Betriebssystemen angeboten und auch unter MAC-OS, Linux mit PaX ASLR und auch iOS oder auch Android sind ASLR-Varianten im Einsatz.
Beispielsweise kann unter Windows Vista, bzw. Windows 7 für eine ausführbare Binärdatei (.exe) oder auch eine DLL (.dll) durch das Setzen eines speziellen Bits in dem PE Header des Executables [7] die Verwendung der ASLR Technik angefordert werden. Für das zu ladende Executable wird in diesem Fall ein spezieller Offset-Wert berechnet, um dynamisch, aus Sicht eines Angreifers randomisiert, die Speicherposition zu bestimmen. Der Offset ist ein Wert aus dem Wertebereich von 1 bis 256 und ist an 64 KB ausgerichtet (aligned). Wird ein Programm ausgeführt, das ASLR Bit gesetzt hat, wird auch das Speicherlayout des Prozesses randomisiert, indem seine Stack- und Heap-Segmente randomisiert angelegt werden und auch der initiale Stack-Pointer durch einen randomisierten Wert aus dem Wertebereich 1 bis 16384 dekrementiert wird. Das Speicherlayout der Stack- und Heap-Segmente wird bei jeder Programmausführung erneut randomisiert festgelegt, während das Layout des Code- und Datensegments sich nur bei einem Re-Boot ändert. Der Offset von gemeinsam genutzten DLLs wird über eine systemglobale Variable zum Bootzeitpunkt ermittelt und die DLL wird in den gemeinsam genutzten Speicherbereich in den Bereichen zwischen $0x50000000$ und $0x78000000$ abgelegt. Die DLLs werden zudem in randomisierter Reihenfolge geladen.

Trusted OS

Speziell abgesicherte Varianten von Standardbetriebssystemen, die als Trusted Operating Systems vertrieben werden (z.B. Trusted Solaris[8], Trusted HP-UX), führen so genannte Compartments ein, die Systembereiche gegeneinander abschotten, so dass sich die Auswirkungen von Buffer-Overflow-Angriffen auf ein Compartment beschränken lassen.

[7] PE (Portable Executable) beschreibt ein Binärformat ausführbarer Programme.
[8] Trusted Solaris wurde 2006 als eigenständiges Produkt eingestellt.

Abschließend sei noch einmal darauf hingewiesen, dass die Buffer-Overflow-Verwundbarkeit sich nicht auf den Stack beschränkt, sondern Heap-Bereiche in gleicher Weise betroffen sind. Auch beschränken sich die heutigen Angriffe bei Weitem nicht nur auf die gängigen Operationen zum Kopieren von Zeichenketten, sondern nutzen eine Vielzahl weiterer Angriffsflächen aus. Wir haben bereits auf die Loop-Probleme, Integer-Overflows oder auch die Unicode Probleme hingewiesen. Alle diese Schwachstellen rühren von Programmierfehlern her, die man zum Teil frühzeitig durch statische Programmanalysen und durch eine sorgfältige Programmierung in den Griff bekommen könnte. Auf diesem Gebiet sind zurzeit eine Reihe von Tool-Entwicklungen in Arbeit, z.B. auch in den Forschungsbereichen von Microsoft, so dass die Hoffnung besteht, dass Buffer-Overflows in Zukunft nicht mehr länger die „Schwachstellen des Jahrzehnts“sind, wie sie noch in den 90ger Jahren von Bill Gates bezeichnet wurden.

2.3 Computerviren

Ein Virus (deutsch *giftiger Saft*) bezeichnet in der Biologie einen Mikro-Organismus, der auf eine lebende Wirtszelle angewiesen ist, keinen eigenen Stoffwechsel besitzt und fähig ist, sich zu reproduzieren. Diese Eigenschaften sind direkt auf Computerviren übertragbar.

2.3.1 Eigenschaften

Der Begriff des Computervirus wurde zum ersten Mal 1985 durch Arbeiten von F. Cohen (u.a. [48]) der breiten Öffentlichkeit bekannt.

Definition 2.1 (Computervirus)

Virus

Ein Computervirus ist eine Befehlsfolge, die ein Wirtsprogramm zur Ausführung benötigt. Viren sind zur Reproduktion fähig. Dazu wird bei der Ausführung des Virus eine Kopie (Reproduktion) oder eine modifizierte Version des Virus in einen Speicherbereich, der diese Befehlssequenz noch nicht enthält, geschrieben (Infektion). Zusätzlich zur Fähigkeit zur Reproduktion enthalten Viren in der Regel einen Schadensteil. Dieser kann unbedingt oder bedingt durch einen Auslöser aktiviert werden.

□

mutierend

Mit den Festlegungen von Definition 2.1 ist klar, dass ein Computervirus kein selbständig ablauffähiges Programm ist. Falls ein Virus sich nicht identisch reproduziert, also keine identische Kopie erzeugt, sondern die den Virus definierende Befehlssequenz modifiziert, so spricht man von einem mutierenden Virus.

Reproduktionsfähigkeit

Zur Codierung der Befehlsfolge eines Virus verwendet man Maschinensprachen ebenso wie Kommandosprachen, Skriptsprachen oder auch Hochsprachen. Potentielle Speicherbereiche für Viren sind der Code ausführbarer Programme von Benutzern, Bereiche des Betriebssystems oder auch Sektoren eines Hintergrundspeichermediums. Der allgemeine Aufbau eines Virus ist in Abbildung 2.4 skizziert. Ein Virus besteht aus einer Viren-Kennung sowie optional aus einem Infektions-, einem Schadens- und einem Sprungteil. Durch die Ausführung des Infektionsteils kopiert sich ein Virus in einen Speicherbereich. Handelt es sich dabei um den in einer Datei gespeicherten Code eines ausführbaren Programms, so werden i.d.R. die Strukturinformationen des infizierten Programms durch die Angabe der neuen Dateilänge sowie durch die Veränderung der Einsprungadresse angeglichen. Die neue

Struktur

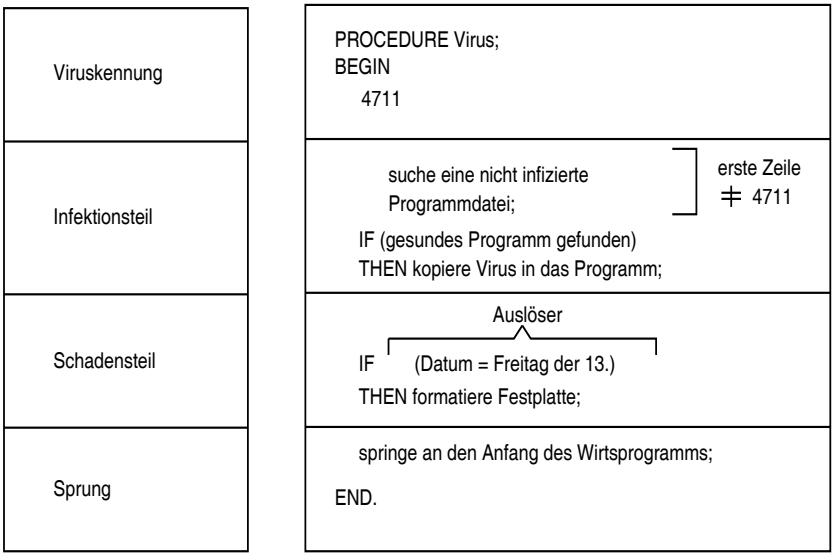

Abbildung 2.4: Allgemeiner Aufbau eines Virus und Beispiel

Einsprungadresse entspricht der Anfangsadresse des Virus-Codes. Soll nach der Ausführung des Virus doch noch das Programm mit der ursprünglichen Funktionalität ausgeführt werden, so enthält der Virus-Code im optionalen Sprungteil eine Rücksprungadresse, die die Einsprungadresse in das ursprüngliche Programm ist.

Anhand einer speziellen Viren-Kennung (z.B. des Wertes `4711` in der Abbildung 2.4), kann ein Virus in seiner Infektionsphase erkennen, ob ein

Kennung

Programm schon vom selben Virus befallen ist. Dies verhindert das wiederholte Infizieren von Programmen. Diese Viren-Kennung (oder eine andere typische Zeichenfolge) dient im Gegenzug Virenerkennungsprogrammen, den Viren-Scannern, zur Aufdeckung von infizierten Programmen.

Bedingung

Die Ausführung des Schadensteils kann von Randbedingungen abhängig gemacht werden. Im Beispiel von Abbildung 2.4 wird auf ein spezielles Datum, nämlich Freitag den 13ten, gewartet. Ist zum Zeitpunkt der Ausführung des Wirtsprogramms die Bedingung erfüllt, so wird die angegebene Schadensfunktion wie die Neuformatierung der Festplatte ausgeführt. Soll nach der Abarbeitung der Befehlssequenz des Virus das ursprüngliche Programm ausgeführt werden, so wird ein Sprungbefehl eingefügt, mit dem an die Anfangsadresse des Programms gesprungen wird.

Durch Viren treten in erster Linie Bedrohungen der Integrität und Vertraulichkeit auf.

2.3.2 Viren-Typen

Erste Generation

Mit dem Einzug des Personal Computers (PCs) in den 80er Jahren in Behörden, Firmen und in Privathaushalte erfolgte eine Abkehr von zentral, meist noch manuell administrierten Großrechenanlagen (engl. *mainframe*) hin zu einer dezentralen Verwaltung isoliert betriebener PCs durch deren Benutzer. Das heißt, die entsprechenden Benutzer übernahmen Aufgaben der Systemadministration und waren berechtigt, Software auf ihren PCs zu installieren. Zusammen mit der steigenden Anzahl gemeinsam genutzter Programme, insbesondere Spielprogrammen und Shareware, war damit der Boden zur Verbreitung von Viren geebnet. Die Viren der ersten Generation verbreiteten sich in erster Linie über das Kopieren von Daten von Diskette zu Diskette sowie über die meist noch manuelle Installation von Programmen auf einzelnen PCs.

Viren-Verbreitung

Isolierung

Durch ein Nachbilden der ursprünglichen, zentralistischen Mainframe-Administration kann jedoch in solchen Umgebungen die Verbreitung von Viren relativ einfach und wirkungsvoll eingegrenzt werden. Dazu ist ein zentraler Server-Rechner so zu konfigurieren, dass jedes neue Softwareprodukt zunächst auf diesem Rechner installiert und getestet wird. Der Server stellt somit eine Art „Quarantänestation“ für Software dar. Das systematische Testen und Kontrollieren von Software muss jedoch einem ausgewählten, vertrauenswürdigen Personenkreis vorbehalten bleiben, damit die Kontrolle nicht unterlaufen wird bzw. nicht durch eine unsachgemäße Behandlung Sicherheitslücken bestehen bleiben. Bemerkenswerter Weise gehört das Einrichten von Quarantäne-Netzen und Quarantäne-Rechnern heute wieder zu den häufig verwendeten Schutz-Maßnahmen in Unternehmen, um in einer dezentralen IT-Landschaft über zentral administrierte Komponenten neue

Software zunächst in einem beherrschten und kontrollierbaren Umfeld zu evaluieren, bevor sie auf Rechner im Unternehmen verteilt werden.

Die häufigsten Viren, die in dieser Zeit auftraten, waren die so genannten Programm- und die Bootsektor-Viren.

Programm-Viren

Link-Virus

Ein Programm-Virus (auch bekannt als Link-Virus) kopiert sich in eine ausführbare Datei. Er wird durch den Aufruf des Programms verbreitet, da mit der Programmausführung auch der Virus-Code ausgeführt wird. In Abbildung 2.5 ist die Struktur eines infizierten Programms skizziert. Nach der Infektion liegt ein ausführfähiges Programm vor, das beim Aufruf zunächst den Virus und dann erst das eigentliche Programm ausführt. Zur Verschleierung des Virus wurden auch die Strukturdaten der infizierten Datei verändert.

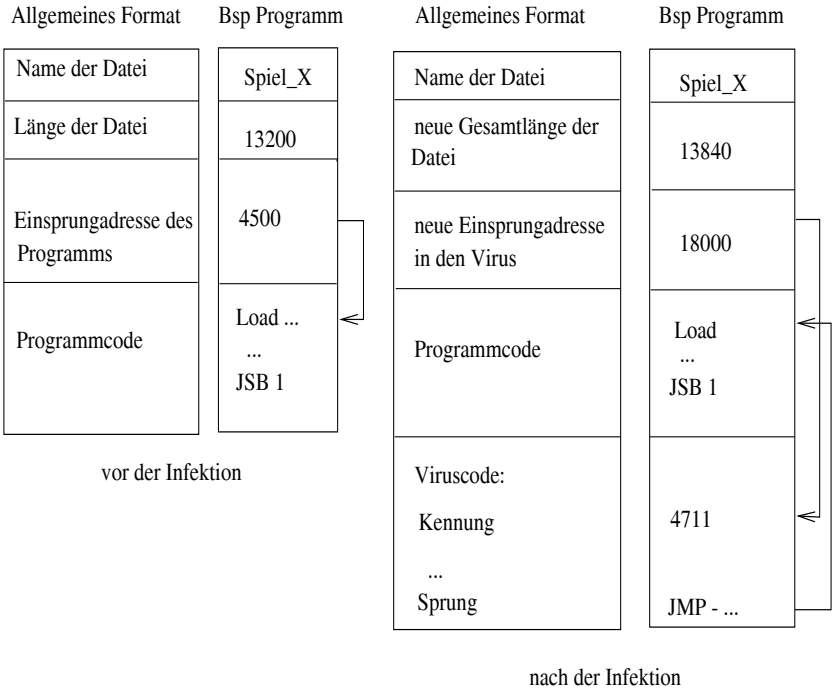

Abbildung 2.5: Infiziertes Programm

Boot-Viren

Boot-Virus

Boot- oder Bootsektor-Viren befallen die Bereiche einer Diskette oder einer Festplatte, deren Daten beim Hochfahren des Rechners, also beim Systemstart, gelesen und in den Hauptspeicher geladen werden. Beim Booten (vgl.

auch Seite 623) eines Rechners initialisiert sich zunächst die CPU und startet einen Code zur Überprüfung der Hardware. Dieser Code befindet sich im ROM BIOS (Basic Input/Output System) auf dem Motherboard. Durch die Ausführung des BIOS-Codes werden im nächsten Schritt die Befehle geladen, die im Bootsektor der Diskette bzw. Festplatte abgelegt sind. Der Bootsektor enthält das Ladeprogramm, durch dessen Ausführung das Betriebssystem in den Hauptspeicher geladen und ausgeführt wird. Ein Boot-Virus wird meist vor dem Bootsektorprogramm in den Speicher geschrieben, so dass bei jedem Hochfahren des Systems zunächst der Virus ausgeführt wird. Bootsektor-Viren sind resident im Hauptspeicher geladen. Die residente Speicherung von Daten bedeutet, dass diese nicht auf den Hintergrundspeicher ausgelagert werden, sondern während der gesamten Arbeitsphase des Systems im Hauptspeicher eingelagert sind. Abbildung 2.6 skizziert einen infizierten Bootsektor. Ein Boot-Virus kann beispielsweise die Interrupt-Handler-Tabelle verändern, die u.a. die Anfangsadresse der Treiberroutinen enthält, die bei einem Geräte-Interrupt auszuführen sind. Konnte beispielsweise anstelle der Anfangsadresse des Treiber-Codes für die Maus der Anfangscode des Virus platziert werden, würde das bedeuten, dass bei einem durch die Maus ausgelösten Geräte-Interrupt stets der Virus-Code anstelle des Handlers für die Maus gestartet würde. Obwohl sie großen Schaden verursachen können, spielen Boot-Viren heutzutage keine nennenswerte Rolle mehr, da sich die Verbreitungswege von Viren und deren Funktionsweise geändert haben.

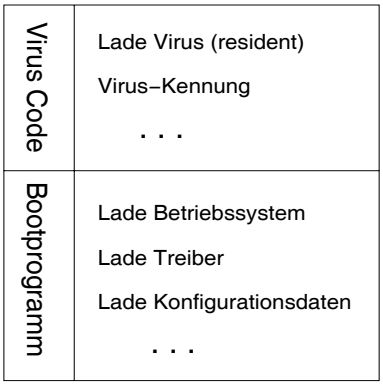

Abbildung 2.6: Infizierter Bootsektor

Viren der zweiten Generation

Durch Übergang zu vernetzten, offenen Systemen eröffnet sich eine Vielzahl von Kanälen, über die fremder Code auf den lokalen Rechner gelangen kann. Beispiele für solche in der Praxis häufig anzutreffende Kanäle sind E-Mails mit Anhang (engl. *attachment*), Java-Applets, elektronische Dokumente oder

auch Bild-Dateien. Auch die Verbesserung heutiger Software-Werkzeuge trägt erheblich zur Verbreitung von Viren bei. Diese zunächst wenig einleuchtende These lässt sich jedoch einfach mit der zunehmenden Automatisierung von Vorgängen bei der Informationsverarbeitung begründen. Wesentliches Ziel der entwickelten Werkzeuge ist es nämlich, möglichst viele Aktivitäten automatisch und transparent für den Benutzer durchzuführen. Diese eigentlich höchst wünschenswerte Eigenschaft hat jedoch für den Virenbereich fatale Konsequenzen. Unter Nutzung von Werkzeugen wie Postscript- und PDF-Interpretern, Textverarbeitungsprogrammen (u.a. Word-Prozessor) oder MIME-Interpretern werden Befehle, die in fremden Dokumenten eingebettet sein können, automatisch ausgeführt. Auf diese Weise werden dann die in den Befehlen enthaltenen Viren transparent für den Benutzer, also ohne dessen explizites Eingreifen, in Umlauf gebracht.

Werkzeuge

Die Verbreitung von Viren über Rechnernetze hinweg hat darüber hinaus auch zu einer Veränderung der Funktionalität von Viren geführt. Während die Viren der ersten Generation vordringlich destruktiv ausgerichtet waren, findet man unter den neueren Viren zunehmend solche, die gezielt einen weiteren Angriff vorbereiten, indem sie sicherheitsrelevante Informationen des Zielrechners sammeln (u.a. abgelegte Passworte), oder indem sie gezielt Konfigurations- und Systemdateien modifizieren. Mit dem Wandel der Schadfunktionen von Viren und deren geänderten Verbreitungswegen verschwimmen auch zunehmend die konzeptuellen Unterschiede zwischen Viren und Würmern (vgl. Abschnitt 2.4). Häufig werden die Begriffe Virus und Wurm deshalb synonym verwendet.

Funktionalität

Makro- oder Daten-Viren

Makros sind in der Regel kleine Code-Teile, die in Daten-Dokumenten eingebettet werden. Solche Makros können beispielsweise bei der Erstellung von Word-Dokumenten oder Tabellenkalkulationen mittels Excel einem Daten-Dokument hinzugefügt werden. Damit erhalten diese Dateien einen ausführbaren Teil, so dass ein Dokument das Programm, durch das sie erstellt wurden, wie z.B. den Word-Prozessor, über die Ausführung eines solchen Makros kontrollieren und das Verhalten des Programms an die spezifischen Anforderungen des Dokuments anpassen kann. Beispielsweise kann man über ein Makro, das einem Word-Dokument hinzugefügt wird, dafür sorgen, dass bei jedem Abspeichern des Dokuments automatisch ein Programm zur Prüfung der Rechtschreibung aufgerufen wird. Makros enthalten Steuerbefehle, die in einer Makrosprache wie zum Beispiel Microsofts Visual Basic for Applications (VBA), formuliert sind und interpretativ ausgeführt werden. Die Erweiterung von reinen Daten-Dateien zu Objekten mit ausführbaren Bestandteilen stellt einen gravierenden Schritt für die Virenproblematik dar. Nun sind auch Daten-Dateien, die bislang nur gelesen und

Makros

nicht ausgeführt wurden, potentielle Wirte für Viren. Dadurch ergibt sich ein sehr großes Potential neuer Bedrohungen. Vom Standpunkt eines Benutzers aus betrachtet reicht jetzt bereits ein lesender Zugriff auf ein Dokument aus, z.B. das Öffnen eines Word-Dokuments, um eine Virus-Verbreitung zu initiieren. Dass sich hinter dem Lese-Zugriff die Ausführung von Makro-Kommandos und damit das Starten eines Virus verbergen kann, ist für den Benutzer nicht unbedingt ersichtlich.

Makro-Virus

Seit 1995 sind die ersten Makro-Viren bekannt. Sie stellen bereits heute die häufigste Ursache für einen Virenangriff dar. Die ersten Vertreter dieser Viren-Klasse traten vorzugsweise in Dateien der Programme Word für Windows (WinWord) sowie in Excel-Programmen auf. Makro-Viren sind in der Regel unabhängig von der verwendeten Rechnerplattform, da sie interpretiert ausgeführt und nicht in Maschinencode übersetzt werden. Diese Plattformunabhängigkeit von Makro-Viren stellt eine weitere gravierende Verschärfung der Viren-Problematik dar, da auf diese Weise die „natürlichen“ Barrieren für die Verbreitung von Viren in einer heterogenen Umwelt entfallen. Die WinWord-Viren haben sich beispielsweise auf PCs mit unterschiedlichen Betriebssystemen u.a. Windows 3.xx, Windows NT oder Windows 95 verbreitet.

Attachment

Daten-Viren treten häufig auch in Postscript-Dateien und in Anhängen (engl. *attachment*) von MIME (Multipurpose Internet Mail Extension) Mails auf. MIME ist eine Erweiterung des SMTP-Protokolls zur Nachrichtenkommunikation, das neben der Übertragung einfacher ASCII-Dateien auch die Übertragung von Grafiken, Sprache etc. ermöglicht. Aus Sicherheitssicht problematisch ist, dass MIME-Nachrichten Befehle beinhalten können, die beim Empfänger der Nachricht ausgeführt werden. So ist es zum Beispiel möglich, beim Empfänger einen Datei-Transfer mittels FTP zu initiieren.

Postscript

Für Viren, die sich in Postscript-Dateien (ps-Dateien) verstecken, gilt ganz Analoges. Postscript ist eine Beschreibungssprache zur Darstellung von Grafikdaten und formatierten Texten. Postscript-Dateien (.ps) werden von einem Interpreter ausgeführt, der bei Postscript-fähigen Druckern direkt in den Drucker integriert ist oder als Programm zur Verfügung stehen muss, um die Darstellung von Postscript-Dateien auf dem Bildschirm zu ermöglichen (z.B. mittels *ghostview*). Die Postscript-Sprache enthält einige Befehle, die durch den Interpreter ausgeführt und für Angriffe missbraucht werden können. Dazu gehören hauptsächlich Befehle zum Löschen oder Umbenennen von Dateien oder zur Erzeugung einer Kommunikationsverbindung, über die Daten ausgetauscht werden können.

ani-Viren

Varianten der Makro-Viren sind Viren, die in Dateien eingebettet werden, die eigentlich keinen ausführbaren Code enthalten, wie beispielsweise

.gif, .wmf oder .ani-Dateien[9]. Die entsprechenden Dateien werden in der Regel über Webseiten oder e-Mails verbreitet. Eine Schwachstelle in den Routinen zur Verarbeitung von Dateien zur Definition des Cursors (.cur), animierter Cursors (.ani) und Icons (.ico) unter Windows 2000, XP, Server 2003 und Vista, kann von einem Angreifer dazu ausgenutzt werden, beliebigen Programmcode einzuschleusen und mit System-Privilegien auf dem Opfer-Rechner auszuführen. Die Schwachstelle besteht darin, dass die Routine zum Laden der Cursor keine Bereichsprüfung durchführt, sondern der Längenangabe, die die Größe des Headers angibt, vertraut. Die Routinen werden z.B. vom Internet Explorer oder von Outlook aufgerufen, wenn sie .ani-Dateien in einer Webseite oder einer HTML e-Mail empfangen. Um die Schwachstelle erfolgreich auszunutzen, ist es also erforderlich, dass das Opfer eine entsprechende Datei auf einer Webseite ansieht bzw. herunter lädt oder eine entsprechende E-Mail-Nachricht öffnet.

Beispiel 2.1 (Sobig Virus)

Sobig

An dem Sobig-Virus, der sich Anfang 2003 rasant ausbreitete, lässt sich die Funktionsweise heutiger Viren sehr gut verdeutlichen. Der Virus nutzte auf schon fast klassische Weise Social Engineering Vorgehen aus, um sich via E-Mail-Attachments zu verbreiten. In der Subjekt-Zeile der Mail traten unverfängliche, die Neugier des Empfängers anregende Bezeichnungen auf, wie *Re: Sample, Re: Document* oder *Re: Movies*. Die eigentliche Mail enthielt eine Nachricht der Art *Please see the attached file for Details*, die den Empfänger dazu animieren sollte, das beigefügte Attachment, das eine mit einem Virus infizierte pif-Datei (z.B. Sample.pif) enthielt, zu öffnen. Durch einen Doppelklick auf den Anhang wurde dessen Infizierungs- und Schadfunktion ausgelöst.

Social Engineering

Infektionsteil

Der aktivierte Virus führte seinen Infektionsteil aus, der zunächst eine Kopie von sich selbst unter dem Namen *Winmgm32.exe* im Windows-Verzeichnis ablegte und folgende zwei Einträge (Registry-Keys) in der Windows-Registry eintrug, so dass der Virus beim Booten automatisch gestartet wird:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows
\CurrentVersion\Run
"WindowsMGM"=C:\WINDOWS\winmgm32.exe
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows
\CurrentVersion\Run
"WindowsMGM" = C:\WINDOWS\winmgm32.exe
```

[9] Animated Cursor Handling

Ferner wurde anhand der Netzwerkfreigabe aktiv nach möglichen Opfersystemen gesucht, um den Virus auf diese Systeme zu kopieren. Durchsucht wurden folgende Verzeichnisse:

```
\WINDOWS\ALL USERS\START MENU\PROGRAMS\STARTUP   oder
 \DOCUMENTS AND SETTINGS\ALL USERS\START MENU\PROGRAMS
                                              \STARTUP
```

Schadensteil

Im Schadensteil führte der Virus verschiedene Teilschritte durch. In einem ersten Schritt verschickte er sich selber in einer E-Mail an alle E-Mailadressen, die u.a. in Adressbüchern des befallenen Rechners zu finden waren. Des Weiteren öffnete er eine jpeg-Datei, die übrigens zu allem Überfluss auch noch ein Pornobild enthielt, und startete damit eine weitere Schadfunktion, die versuchte, ein Trojanisches Pferd (vgl. Abschnitt 2.5) aus dem Netz zu laden. War dies erfolgreich, so wurde das Trojaner-Programm gestartet, das u.a. einen Key-logger (*key-logger.dll*) auf dem befallenen Rechner installierte, um die Passwort-Eingaben von Benutzern abzufangen. Weiterhin spähte das Trojaner-Programm Registry-Einträge aus, um Informationen über die System-Konfiguration des befallenen Rechners zu sammeln. Über eine TCP-Verbindung, die der Trojaner zu der Seite *geocities.com* in periodischen Abständen aufbaute, erhielt er eine vom Angreifer vorab präparierte URL, an die die gesammelten Informationen (u.a. IP-Adresse des befallenen Rechners, Benutzerkennung(en) und Passwort(e)) übergeben wurden. Außerdem versuchte der Trojaner den Port 1180 auf dem befallenen Rechner zu öffnen, wodurch dem Angreifer ein entfernter Zugriff auf den Rechner ermöglicht wurde.

Trojaner

Fazit

Zusammenfassend ist festzuhalten, dass der Sobig-Virus viele klassische Charakteristika aufweist. Seine schnelle und große Verbreitung erfolgte über E-Mail-Attachments, die von sorglosen Benutzern gutgläubig geöffnet wurden (Social Engineering), da die Mail angeblich von einer bekannten E-Mail-Adresse stammte, eine unverdächtige Subjekt-Zeile aufwies und der Mail-Körper auf Details im Anhang verwies und damit die Neugierde des Empfängers weckte. Das Ablegen von Einträgen in der Windows-Registry, um ein automatisches Starten des Virus beim Booten zu erreichen, wie auch die Installation von Trojaner-Programmen auf den befallenen Rechnern, zählen zu den üblichen Schadensroutinen heutiger Viren.
Bereits im August 2003 verbreitete sich erneut äußerst rasant der Sobig.F-Virus, der eine nur geringfügig veränderte Variante des Sobig-Virus darstellte. Nach ganz ähnlichem Muster arbeiteten auch die Viren, die sich im Frühjahr 2004 sehr schnell verbreiteten, wie der netsky-Virus. Als Variante wird der Virus-Code häufig in einem per Passwort-geschützten zip-Archiv versandt und das Passwort erhält der Empfänger in der Mail gleich mitgeliefert. Über die Eingabe des Passwortes und das Entzippen wird dann der Virus gestartet. Diese in sehr kurzen Abständen immer wieder neu

auftretenden Massen-Viren beweisen einmal mehr eindrucksvoll, dass die Erkennungsleistungen heutiger Anti-Viren Software unzulänglich und das Sicherheitsbewusstsein der Nutzer stark verbesserungswürdig ist. Neben der Steigerung des Sicherheitsbewusstseins sind dringend verbesserte Techniken notwendig, um Rechner a priori vor derartiger Schadsoftware zu schützen. Dazu zählen Maßnahmen, um ein Anlegen von Programmkopien (z.B. bei Öffnen von Anhängen) nur in gesicherten Quarantänebereichen zu ermöglichen, und um den unkontrollierten Zugriff auf sensitive Systeminformation (z.B. schreibender/lesender Zugriff auf die Windows-Registry) oder die Ausführung von Systemdiensten (z.B. Aufbau einer TCP-Verbindung) zu unterbinden.

▲

2.3.3 Gegenmaßnahmen

Wir können die Menge der möglichen Gegenmaßnahmen grob in zwei Klasse einteilen, nämlich der präventiven Maßnahmen zur Virenabwehr der Werkzeuge zur Erkennung eines Virenbefalls.

Präventive Maßnahmen

Zu dieser Klasse von Abwehrmaßnahmen zählen wir das Beschränken von Schreibberechtigungen, das Verschlüsseln von Daten, die Verwendung digitaler Fingerabdrücke sowie administrative Maßnahmen zur Abwehr von Makro-Viren. Auf diese Bereiche wird im Folgenden kurz eingegangen.

Rechtebeschränkung

Da heutzutage nahezu alle im Einsatz befindlichen Rechner an das Internet angebunden sind, ist jedes dieser Geräte potentiell gefährdet und erfordert den Einsatz von Überwachungswerkzeugen und Viren-Scannern. Die beschränkte Vergabe von Schreibrechten an Programme ist eine wirksame Abwehrmaßnahme gegen Programmviren, da dadurch das Modifizieren bestehender Programme erschwert und damit die Verbreitung von Viren eingedämmt wird. So benötigen zum Beispiel viele Spielprogramme höchstens Schreibzugriffe auf temporär anzulegende Dateien, so dass auch nur für das `TMP`-Verzeichnis entsprechende Berechtigungen erteilt werden müssen.

Verschlüsseln

Durch die Verschlüsselung von gespeicherten Programmen kann man verhindern, dass ein Virus gezielt in der in Abbildung 2.5 skizzierten Weise in ein Programm integriert wird. Ein einfaches Überschreiben des verschlüsselten Codes durch einen Virus führt nach der Entschlüsselung zu nicht ausführbarem Code. Das Überschreiben zerstört zwar die Quelldatei, aber diese Vorgehensweise unterbindet die Viren-Verbreitung sowie die Ausführung von Schadensteilen von Viren. Die Verschlüsselung von Dateien wird zurzeit bereits durch eine Reihe von Produkten wie dem Encrypting File System (EFS) als Bestandteil von Windows2000/XP (vgl. Abschnitt 12.6.2) unterstützt.

Hashfunktion

Eine weitere Möglichkeit, durch präventive Maßnahmen eine Vireninfektion vor Ausführung eines Programms erkennen zu können, besteht darin, einen digitalen Fingerabdruck des Programms (vgl. Abschnitt 8.1) zu berechnen. Ein digitaler Fingerabdruck (engl. *digest*) ist ein Bitstring fester Länge, der das Programm eindeutig beschreibt. Diese Bitstrings sind schreibgeschützt zu verwalten, so dass vor der Ausführung eines Programms dessen Fingerabdruck zunächst erneut berechnet und mit dem ursprünglichen Wert verglichen werden kann. Da der Fingerabdruck einen Programmcode eindeutig beschreibt, führt die Veränderung des Codes nach der Integration von Virus-Code zu einem neuen Bitstring als Fingerabdruck des modifizierten Programms. Diese Abweichung zwischen dem ursprünglichen und dem aktuellen Fingerabdruck ist vor einer Programmausführung erkennbar. Auf diese Weise kann die Verbreitung des Virus unterbunden, der erstmalige Befall jedoch nicht verhindert werden. Zur Behebung eines Virenbefalls ist eine gesicherte Version von einem Backup-Medium wiedereinzuspielen, wobei wiederum die digitalen Fingerabdrücke überprüft werden müssen, um sicherzustellen, dass keine infizierte Version eingespielt wird.

Quarantäne

Zur Abwehr von Makro-Viren ist zu verhindern, dass diese beim Lesen eines Dokumentes durch die zum Lesen verwendeten Werkzeuge (u.a. Word-Prozessor, Postscript-Interpreter) automatisch ausgeführt werden. Da eine automatische Meldung, ob ein zu ladendes Dokument Makros enthält, nicht immer erfolgt (z.B. erfolgt dies unter Windows erst mit WinWord Version 7.0a), sollte in Umgebungen, die keine automatisch ablaufenden Makros erfordern, die Ausführung solcher Makros unterbunden werden. Die Kontrolle von erweiterten E-Mails oder anderer über das Netz ausgetauschter Dokumente ist nicht immer möglich. Mails bzw. Dokumente von unbekannten Absendern sollten zunächst unter Nutzung von Viren-Scannern überprüft und im Zweifelsfall ungelesen gelöscht oder nur in einer isolierten Ausführungsumgebung gelesen werden. So stehen beispielsweise unter Windows zur automatischen Überprüfung von Dokumenten Suchprogramme zur Verfügung, die periodisch im Hintergrund ablaufen und somit die laufende Anwendung kaum behindern.

Isolierung

Werkzeuge zum Antivirenmanagement

Scanner

Die am häufigsten eingesetzten Werkzeuge gegen Viren sind die Viren-Scanner. Bekannte Viren besitzen eine Viren-Kennung oder enthalten spezifische Bytemuster bzw. Codesequenzen, woran sie erkannt werden können. Viren-Scanner verwalten umfangreiche Datenbanken mit Kennungen bekannter Viren sowie mit Regeln, die u.a. festlegen, welche Dateien bzw. Dateitypen (z.B. alle ausführbaren Programme, u.a. .exe oder .com-Dateien) auf Viren zu überprüfen sind. Bei einem Viren-Scan werden die gespeicherten Dateien auf das Vorhandensein von Viren-Kennungen untersucht und

es werden die infizierten Objekte sowie die gefunden Viren angezeigt. Auf diese Weise können natürlich nur bereits bekannte Viren erkannt werden. Mutierende Viren oder neue Viren schlüpfen in der Regel durch das Netz dieser Scanner.

Heuristik

Aus diesem Grund versuchen einige Erkennungsprogramme anhand von Heuristiken auch unbekannte Viren bzw. Abwandlungen bekannter Viren zu entdecken. Die Heuristiken suchen nach verdächtigen Codesequenzen in Programmen, die auf Virenaktivitäten hindeuten, wie zum Beispiel Befehle, die dem Suchen nach neuen Opfern und dem Kopieren/Replizieren des Virus entsprechen. Die anhaltenden Probleme mit Viren-verseuchten Rechnern verdeutlichen jedoch, dass diese Heuristiken unzureichend sind, da die Anzahl der möglichen Veränderungen existierender Viren sehr groß ist und durch Heuristiken unmöglich alle Mutationen bekannter Viren abgedeckt werden können. In Anbetracht der rasanten Entwicklung und Verbreitung neuer Viren ist deshalb auf jeden Fall eine regelmäßige Aktualisierung der Datenbanken mit Viren-Kennungen unerlässlich.

Aktivitätskontrolle

Ein anderer Ansatz zur Bekämpfung von Viren wird durch Werkzeuge verfolgt, die eine Aktivitätskontrolle durchführen. Dazu werden Programme während ihrer Ausführung überwacht und auf Verhaltensweisen untersucht, die für einen Virenbefall typisch sind. Verdächtige Aktionen umfassen den wiederholten, modifizierenden Zugriff auf ausführbare Dateien, das Suchen nach ausführbaren Dateien oder Versuche, direkt auf externe Speichermedien zuzugreifen. Derartige Tools können als sehr einfache Varianten von Einbruchserkennungs-Systemen (engl. *intrusion detection system IDS*) aufgefasst werden.

Monitoring

Als dritte Klasse von Werkzeugen sind die Monitoring-Tools zu nennen, die Dateien (im Wesentlichen handelt es sich dabei um wichtige Systemdateien) überwachen und durchgeführte Modifikationen anhand geänderter Strukturinformationen oder anhand von Abweichungen bei berechneten digitalen Fingerabdrücken erkennen.

Haftung

Auf Maßnahmen zur Abwehr von Viren wird auch in Zukunft ein Augenmerk bei der Sicherheit von IT-Systemen liegen müssen. Solange jedoch Softwarehersteller und Benutzer nicht in die Pflicht dafür genommen werden können, dass sie ihre Software mit mangelhaften Sicherheitsniveaus ausstatten bzw. ihre Systeme unsicher vorkonfigurieren und fahrlässig mit potentiellen Schadprogrammen umgehen, werden weltweite Virenvorfälle immer wieder auftreten. Erste Ansätze, unachtsamen Anwendern, von deren PCs sich Viren verbreitet haben, mit Klagen auf Schadensersatz wegen der für die Beseitigung der Schäden verursachten Kosten zu drohen, haben aber zurzeit wohl noch keine sehr große Aussicht auf Erfolg. Trotzdem erscheint

das Vorgehen, über Fahrlässigkeitsklagen Haftungsansprüche durchzusetzen und Schaden sowie Verantwortung zu teilen, sehr sinnvoll, um in Zukunft von Unternehmen und Einzelpersonen eine größere Verantwortung gegenüber Dritten erwarten zu können. Eine allgemeine rechtliche Verpflichtung zur Vorsorge mit entsprechenden Haftungsansprüchen erscheint angesichts der großen Verbreitung von Viren durchaus möglich. Beim heutigen Kenntnisstand über die Virenproblematik muss es deshalb schon fast als Fahrlässigkeit angesehen werden, wenn auf einen Virenschutz verzichtet wird.

2.4 Würmer

Im Gegensatz zu einem Virus ist ein Wurm ein eigenständiges, ablauffähiges Programm. Das bedeutet, dass ein Wurm kein Wirtsprogramm zur Ausführung benötigt.

Definition 2.2 (Wurm)

Wurm

Ein Wurm ist ein ablauffähiges Programm mit der Fähigkeit zur Reproduktion. Ein Wurm-Programm besteht in der Regel aus mehreren Programmteilen, den Wurm-Segmenten. Die Vervielfältigung erfolgt selbstständig meist unter Kommunikation mit anderen Wurm-Segmenten.

□

Verbreitung

Die Verbreitung von Würmern erfolgt insbesondere über ein Netzwerk, indem sich der Wurm auf andere Rechner innerhalb des Netzes kopiert. Ausgangspunkt für einen Wurmangriff sind häufig Systemprozesse, die ständig rechenbereit sind oder in regelmäßigen Abständen aktiviert werden. Würmer bedienen sich dieser Prozesse, um beispielsweise über eine Buffer-Overflow-Attacke in das Opfersystem zu gelangen. So nutzte im Jahr 2001 der Code Red Wurm einen Pufferüberlauf im Microsoft Internet Information Service (IIS) aus, um sich zu verbreiten. Die ausführbaren Programmteile eines Wurms können den Quellcode eines Programms beinhalten, der auf dem zu befallenden Rechner übersetzt werden muss, sie können aber auch direkt in ausführbarer Maschinensprache oder in interpretierbarer Sprache, z.B. Shell-Kommandos, geschrieben sein. Bei einem Pufferüberlauf-Angriff auf einen Systemprozess werden diese Befehle dann sogar im privilegierten Modus des Betriebssystemkerns ausgeführt, d.h. der Angreifer erlangt unbeschränkte Zugriffsberechtigungen.

Bedrohungen

Durch Würmer treten ebenso wie durch Viren Bedrohungen der Integrität und Vertraulichkeit, aber auch Bedrohungen der Verfügbarkeit auf, die so genannten Denial-of-Service Angriffe. Würmer beanspruchen in der Regel

viele Ressourcen, da sie bei ihrer Verbreitung über das Netzwerk häufig eine sehr hohe Netzlast erzeugen und durch einen möglichen Mehrfachbefall von Rechnern deren Speicherressourcen ausschöpfen können.

Die ersten Wurm-Programme wurden 1979 im Xerox Palo Alto Research Center als sinnvolle und nicht bedrohliche Anwendungen entwickelt, um eine verteilte Berechnung wie zum Beispiel das Komprimieren von Daten durchzuführen. Einer der ersten und bekanntesten Würmer mit einer bedrohlichen Funktionalität war der so genannte Internet-Wurm.

Beispiel 2.2 (Internet-Wurm)

Internet-Wurm

Am Abend des 2. November 1988 wurde der Internet-Wurm (u.a. [197], RFC1135) gestartet. Der Wurm verbreitete sich im Verlauf der Nacht über das Internet, und als er am nächsten Morgen gestoppt wurde, hatte er bereits 6000 Rechner befallen.

genutzte Schwachstellen

Der Wurm nutzte einige bekannte Fehler und Eigenheiten von Unix aus, die mittlerweile in den meisten Systemen behoben sein sollten. Zu den allgemeinen, auch in heutigen Systemen noch vorhandenen Schwachstellen, die indirekt zur Verbreitung des Wurms beitrugen, zählt, dass Dienstprogramme und ihre Hintergrundprozesse keinem bestimmten Benutzer zugewiesen werden, so dass sich ein Angreifer ihrer bedienen kann.

Angriff

Das Ziel des Internet-Wurms war es, auf einem fremden Ziel-Rechner eine Shell zu starten, um damit das Wurmprogramm auf diesem Rechner zu laden, dort zu übersetzen und auszuführen. Um eine solche Shell zu starten, wurden drei verschiedene Angriffsversuche durchgeführt: (1) Mit einem „Brute force“ Angriff wurde versucht, Passworte auf dem Ziel-Rechner zu knacken (Remote Shell Angriff).

remote shell

sendmail

(2) Falls der Ziel-Rechner eine bekannte Sicherheitslücke im `sendmail` Programm enthielt, wurde diese ausgenutzt, um ein Programm zu senden, das auf dem Zielrechner ausgeführt wurde, um eine Shell zu starten. Die Entwickler des `sendmail` Kommandos hatten zum Debuggen einen speziellen Debug-Modus eingeführt, durch den eine Nachricht zu einem Programm und nicht zum Mailer gesendet wurde. Falls der Ziel-Rechner es ermöglichte, diesen Modus einzustellen, wurde er vom Wurm durch das Versenden einer dediziert konstruierten Befehlsfolge ausgenutzt. Die Ausführung dieser Befehlssequenz bewirkte, dass auf dem Ziel-Rechner eine Shell gestartet wurde, mit der das Wurmprogramm übersetzt und ausgeführt wurde. Da mit dem Debug-Modus Superuser-Rechte verknüpft waren, konnte sogar eine Shell mit Root-Rechten erzeugt werden.

fingered

(3) Als Drittes wurde versucht, einen Implementierungsfehler im `fingered` Programm auszunutzen. Mit diesem Programm kann man Informationen über Benutzer eines Systems erfragen. Das `fingered` Programm nutzt die Systemroutine `gets()`, die eine Eingabe in einen String kopiert. In alten Versionen dieses Programms fehlte eine Bereichsprüfung, so dass mit einer Buffer-Overflow Attacke der Puffer der `gets`-Routine durch einen zu langen Eingabestring zum Überlaufen gebracht werden konnte. Der Wurm nutzte diese Lücke aus, indem er gezielt einen langen Eingabestring erzeugte und Maschinenbefehle auf dem Systemkeller ablegte, durch deren Ausführung eine Shell auf dem befallenen Rechner gestartet wurde. Damit konnte der Angreifer auf dem fremden Rechner Kommandos ausführen.
▲

CERT

Als Reaktion auf den Wurm gründete die U.S. Defense Advanced Research Projects Agency (DARPA) das erste Computer Emergency Response Team (CERT), das die Aufgabe hat, sich mit Sicherheitsfragen rund um das Internet zu beschäftigen.

ILOVEYOU

Im Mai 2000 sorgte der ILOVEYOU-Wurm für einiges Aufsehen, da er sich in einer bis dahin nicht bekannten Geschwindigkeit verbreitete und zum Zusammenbruch der elektronischen Datenverarbeitung in vielen europäischen und amerikanischen Firmen und Behörden führte. Man schätzt den verursachten Schaden durch Datenverluste und Produktivitätsausfall weltweit auf über 8.76 Millionen US Dollar.

Beispiel 2.3 (ILOVEYOU)

Bei dem ILOVEYOU-Wurm handelte es sich um einen Wurm, der als .vbs-Datei (Visual Basic Script) über E-Mail Attachments verbreitet wurde, in deren Betreff-Zeile ILOVEYOU stand. Die Verbreitung des Wurms setzte eine Microsoft Windows-Umgebung voraus, da der Wurm sich der Daten von Windows Outlook bediente, um eine Mail mit dem Wurm als Attachment an die in der Adressdatei gespeicherten Mailadressen zu senden. Der Wurm zerstörte gezielt Dateien des lokalen Dateisystems, die u.a. JPEG-Bilder, MP2- oder MP3 Musik-Daten oder Videodateien enthielten. Schließlich durchsuchte der Wurm auch die lokale Festplatte nach Passworten und versuchte, eine Verbindung aufzubauen, um diese Daten an den Programmierer des Wurms zu übermitteln.
▲

Nahezu wöchentlich treten neue Wurm-Varianten in Erscheinung, die häufig nach einem ähnlichen Angriffsschema ablaufen. Als Beispiel hierzu wird abschließend kurz auf den Blaster- bzw. Lovesan-Wurm eingegangen.

Beispiel 2.4 (Lovesan- bzw. Blaster-Wurm)

Lovesan

Im Sommer 2003 verbreitete sich der so genannte Lovesan-Wurm[10] äußerst rasant und richtete beträchtliche Schäden an. Das Besondere an diesem Wurm war, dass er nicht nur Server-Rechner sondern insbesondere auch Heim-PCs zum Angriffsziel hatte. Sein Angriffsvorgehen basiert jedoch wiederum auf Standardangriffstechniken, die im Folgenden kurz erläutert werden.

Angriffsziel

Das eigentliche Angriffsziel war der Server *windowsupdate.com*, der Patches für Windows-Systeme bereitstellt, die von Windows-Benutzern via Windows-Update auf den eigenen Rechner herunter geladen werden können. Der Blaster-Wurm verursachte einen Distributed Denial of Service (DDoS) Angriff auf diese Web-Site. Mit diesem Angriff wurden systematisch Opfer-Systeme vorbereitet, die nach dem Stichtag 16.08.2003 SYN-Anfragen an den Port 80 des Update-Servers senden sollten. Es handelte sich hierbei um eine SYN-Flood Attacke mit gespooften Absenderinformationen, wie sie auf der Seite 123 beschrieben wird. Die gespooften Absenderadressen waren hier zufällig gewählte IP-Adressen eines Klasse B Netzes. Durch den DDoS-Angriff sollte der Update-Server so überlastet werden, dass die Benutzer keine Patches mehr herunterladen konnten. Um dies zu vermeiden hatte Microsoft, nach bekanntwerden des drohenden DDoS-Angriffs vorsorglich den DNS-Eintrag *windowsupdate.com* entfernen lassen.

Buffer-Overflow

Zur Angriffsvorbereitung nutzte der Wurm eine bekannte Schwachstelle aus, nämlich eine Buffer-Overflow Verwundbarkeit (vgl. Abschnitt 2.2) im DCOM RPC-Dienst an TCP-Port 135 unter dem Windows-Betriebssystem. Um Systeme zu finden, die einen derartigen Angriffspunkt aufweisen, scannte der Wurm entweder das lokale Klasse C-Subnetz oder wählte einen beliebigen IP-Adressbereich aus, um in diesem Adressbereich Systeme mit geöffnetem Port 135 (oder auch Port 139, 445 oder 593) zu identifizieren. An den Port 135 eines derart als mögliches Opfer identifizierten Rechners sendete er sodann ein präpariertes TCP-Paket. Ausgenutzt wurde dabei, dass TCP-Pakete, die an Port 135 empfangen werden, ungeprüft an den DCOM-RPC Dienst weitergeleitet wurden. Das präparierte Datenpaket verursachte einen Buffer-Overflow, wodurch ausführbarer Code auf das Opfersystem gelangte und gleichzeitig den RPC-Dienst zum Absturz brachte sowie einen Reboot des Systems verursachte.

remote Shell

Der gezielte Buffer-Overflow Angriff gewährte dem Angreifer Superuser-Rechte bei der Ausführung des eingeschleusten Codes. Diese wurden genutzt, um eine entfernte Shell (remote Shell) an TCP-Port 444 des Opfer-Rechners zu starten. Über diese war es dann in weiteren Schritten möglich, eine TFTP-Verbindung über den UDP-Port 69 zum Ausgangsrechner der

[10] auch unter dem Namen Blaster-Wurm bekannt

Wurm-Attacke aufzubauen sowie die Datei *msblast.exe* in das Windows-Systemverzeichnis des Opferrechners herunter zu laden und zu starten. Mit der Ausführung von *msblast.exe* wurde ein Windows-Registry-Eintrag beispielsweise der folgenden Art

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
\CurrentVersion\Run
windows auto update= msblast.exe I just want to say
                                    LOVE YOU SAN!! bill
```

erzeugt[11]. Der Blaster-Code prüfte das aktuelle Datum des Opfersystems und versendete beim Erreichen bzw. Überschreiten des Datums 16.08.03 die oben bereits angesprochenen 40 Byte SYN-Pakete mit einer gefälschten Absender-IP-Adresse an den Rechner *windowsupdate.com*.

▲

Fazit

Der Lovesan- bzw. Blaster-Wurm ist ein typisches Beispiel eines Wurmes, der einen Programmierfehler in einem Betriebssystemdienst ausnutzt (Buffer-Overflow), um mit Superuser-Privilegien einen Kommandointerpreter (Shell) auf dem Opfersystem zu starten, TCP- und/oder UDP-Verbindungen aufzubauen und darüber beliebigen weiteren Code auf das Opfersystem herunter zuladen. In diesem speziellen Fall wurden die Opferrechner „nur“ für einen DDoS-Angriff instrumentalisiert, aber natürlich könnten sich auf diesem Weg genauso auch Angriffe auf die Integrität und Vertraulichkeit der Daten auf dem Opferrechner selber ergeben. Einmal mehr zeigt das Beispiel, dass eine sorglose Konfiguration mit z.B. zu vielen geöffneten Ports leicht auszunutzende Angriffsflächen bietet.

Das erfolgreiche Wirken von Würmern wie dem Blaster-Wurm wird durch Design- und Implementierungsfehler in Systemdiensten ermöglicht. Davon sind Open Source Betriebssysteme ebenso betroffen wie proprietäre Betriebssysteme, wie die Systeme der Windows-Familie. Letztere stehen aber stärker im Visier, da sie viel größere Verbreitung besitzen, so dass die Betriebssystem-Monokultur auch die Ausbreitung von Würmern fördert. Mangelnde Sicherheitsmaßnahmen in diesen Systemen eröffnet Programmen nahezu unbeschränkte Zugriffe auf Daten der lokalen Festplatte. Daneben verdeutlichen die rasante Verbreitung von Würmern und die sehr hohe Wiederholungsrate (Auftreten von Wurmvarianten) erneut die mangelnden Kenntnisse und das mangelnde Sicherheitsbewusstsein heutiger Benutzer.

Gegenmaßnahmen

Patch

Im Gegensatz zu Viren, die sich meist auf „legalem“ Weg verbreiten, versuchen Würmer Bugs und Lücken in System- und Anwendungssoft-

[11] Daher rührt der Name Lovesan-Wurm.

ware auszunutzen. Da auf den einschlägigen WWW-Seiten (siehe u.a. http://www.securityfocus.com) und Mailinglisten sehr schnell aktuelle Informationen über ausgenutzte Sicherheitsschwachstellen veröffentlicht und auch Patches zum Beheben der Schwachstellen angeboten werden, sollte sich jeder zuständige Systemadministrator hierüber informieren und seine Systeme absichern. Die Lücken in den `sendmail` bzw. `fingered` Programmen waren beispielsweise schon lange bekannt, bevor der Internet-Wurm sie ausnutzte. Aktuelle Informationen über Lücken und Problembereiche werden regelmäßig von den CERTs veröffentlicht.

Vielfach lassen sich mögliche Einfallstore für Würmer bereits durch eine geeignete Konfigurierung mit einem eingeschränkten Diensteangebot (z.B. Schließen von nicht benötigten Ports) verkleinern oder schließen.

minimale Rechte

Durch eine restriktive Vergabe von Zugriffsberechtigungen, insbesondere auch von Leseberechtigungen, lässt sich ein unerlaubtes Akquirieren von Informationen und das Einbringen von fremdem Code beschränken. Besonders Passwortdaten sind so zu schützen, dass nicht jeder Benutzer Leserecht darauf erhält. Generell sind differenzierte und restriktive Rechtefestlegungen und Kontrollen notwendig, insbesondere bei Zugriffen von entfernten Rechnern. Die fehlende differenzierte Zugriffskontrolle in den Windows Betriebssystemen (Windows 95, 98) hat ursächlich dazu beigetragen, dass sich der *ILOVEYOU*-Wurm in einer bis dahin noch nicht bekannten Geschwindigkeit ausbreiten konnte.

2.5 Trojanisches Pferd

Sage

Der Begriff des Trojanischen Pferdes geht zurück auf die Sage vom Kampf um die Stadt Troja, in der die Griechen nach 10-jährigem Kampf als Zeichen des Rückzuges den Trojanern ein großes, hölzernes Pferd zum Geschenk vor die Tore der belagerten Stadt stellten. Die siegestrunkenen Trojaner zogen das Pferd in das Innere der Stadtmauern, um sich an ihrem Geschenk zu erfreuen und um ihren vermeintlichen Sieg zu feiern. Im Verlauf der Nacht offenbarte das Pferd sein verstecktes Geheimnis; es verbarg griechische Soldaten in seinem Inneren, die die Tore der Stadt öffneten, so dass sie von den Griechen eingenommen werden konnte.

2.5.1 Eigenschaften

Das Trojanische Pferd in der Sage beschreibt die Charakteristika von Programmen, die als Trojanische Pferde bezeichnet werden, sehr genau: Es wird eine Funktionalität vorgetäuscht, die Vertrauen erweckt, die aber durch eine verborgene Schadens-Funktionalität ergänzt wird.

Definition 2.3 (Trojanisches Pferd)

Trojanisches Pferd

Ein Trojanisches Pferd (auch Trojaner genannt) ist ein Programm, dessen implementierte Ist-Funktionalität nicht mit der angegebenen Soll-Funktionalität übereinstimmt. Es erfüllt zwar diese Soll-Funktionalität, besitzt jedoch eine darüber hinausgehende, beabsichtigte zusätzliche, verborgene Funktionalität.

□

Unter die Definition 2.3 fallen somit keine Programme, die infolge von Programmierfehlern ein von der Soll-Funktionalität abweichendes Verhalten zeigen, sondern nur solche, für die diese Erweiterungen absichtlich integriert wurden. Ein Trojanisches Pferd besitzt verborgene Eigenschaften, um z.B. in ein System einzudringen, um Daten aufzuzeichnen oder zu manipulieren. Auf Benutzerebene wird dabei korrektes Verhalten vorgetäuscht. Trojanische Pferde können mit dem Programmstart aktiviert oder durch spezielle Auslöser, wie beispielsweise das Eintreten eines bestimmten Datums, gestartet werden. Man spricht hierbei auch häufig von einer logischen Bombe.

logische Bombe

Trojanische Pferde können in ganz unterschiedlichen Bereichen eines Systems auftreten. Beispiele dafür sind Editoren oder Textverarbeitungsprogramme, die die Inhalte edierter Dateien unbemerkt und unautorisiert kopieren. Weitere Beispiele sind manipulierte Datenbanken, durch die sensible Informationen zum Angreifer durchsickern können, oder auch manipulierte Betriebssystemfunktionen, durch die der Angreifer beispielsweise zusätzliche Zugriffsrechte erlangt.

Beispiel 2.5 (Trojanische Pferde)

Zinsberechnung

Eines der ersten bekannt gewordenen Trojanischen Pferde trat in einer Bank auf. Ein Angestellter hatte den Auftrag, ein Programm zu schreiben, das Zinsabrechnungen auf drei Stellen genau durchführt. Der Programmierer erfüllte diese Aufgabe und zusätzlich sorgte er dafür, dass Restbeträge, die beim Abrunden anfielen, auf sein Konto gutgeschrieben wurden. Innerhalb kürzester Zeit sammelte er dadurch einen großen Geldbetrag an.

CAD-Demo

Ein weiteres bekanntes Trojanisches Pferd, das 1992 für einiges Aufsehen gesorgt hatte, stammte von der Firma CadSoft, einer Firma, die CAD-Software entwickelt. Diese Firma verteilte eine Demoversion ihres Produkts. Bei der Ausführung dieses Demoprogramms wurde auch ein Formular erzeugt, das man zur Bestellung eines Handbuches verwenden konnte. Die zusätzliche, verborgene Funktionalität des Demoprogramms bestand darin, dass die Festplatte des Benutzer-PCs nach Programmen der Firma CadSoft durchsucht wurde und auf dem Bestellformular Art und Anzahl der gefundenen Programme codiert wurden. Da sich Benutzer, die im Besitz von

Raubkopien der CAD-Software waren, bevorzugt das Handbuch zuschicken ließen, gelang es, viele widerrechtliche Benutzer zu entlarven. Anzumerken bleibt jedoch, dass das Ausschnüffeln der Festplatte von Benutzern mittels Trojanern rechtlich nicht zulässig ist und im vorliegenden Fall dann auch entsprechend geahndet wurde.

Gefälschte Rechnungen

Im Jahr 2007 wurden Trojanische-Pferd Programme sehr häufig über Massen-E-Mails verbreitet, die sich als Rechnungen zum Beispiel der Partnervermittlung single.de oder des Versandhauses Quelle tarnen. Der Mail-Empfänger erhält eine Zahlungsaufforderung mit dem Hinweis, dass die Details im beigefügten Anhang nachzulesen seien. Diese Anhänge sind in der Regel .zip oder .rar Archive, die ein Trojanisches Pferd enthalten, das beim Öffnen durch den Benutzer den Rechner infiziert und ggf. auch direkt weiteren Schadcode nachlädt.

▲

Trend

Heutige Trojanische Pferd-Programme bieten den Angreifern eine Vielzahl von Möglichkeiten, den befallenen Rechner zu kontrollieren, gespeicherte Daten auszuspähen, oder aber auch Tastatureingaben, wie zum Beispiel Passwörter, sowie Bildschirmausgaben aufzuzeichnen und über eine Netzwerkverbindung zu versenden. Häufig laden sie automatisch Updates und weiteren Schadcode aus dem Internet auf den befallenen Rechner herunter. Aus dem Lagebericht des BSI zur IT-Sicherheit in Deutschland[12] aus dem Jahr 2007 ist zu entnehmen, dass der Anteil der Trojanischen Pferde unter den Schadprogrammen den Anteil der Viren und Würmer bereits deutlich überholt hat. So waren laut einer Erhebung von Trend Micro im zweiten Quartal des Jahres 2007 83 Prozent der neu entdeckten Schadsoftware Trojanische Pferde. Auch im Bereich der Angriffe mit dem Ziel des Identitätsdiebstahls durch Phishing wächst der Anteil der Trojaner, die diesen Angriff durchführen. Gemäß dem BSI-Lagebericht führen Banken in Deutschland nur noch ca. 10 Prozent der durch Phishing aufgetretenen Schadensfälle auf E-Mail-basierte Angriffe zurück, und machen für 90 Prozent der Schadensfälle Trojanische Pferde verantwortlich.

2.5.2 Gegenmaßnahmen

minimale Rechte

Sensible Daten, wie Passworte, PINs und TANs sollten wenn möglich auf sicheren externen Medien, wie einer Smartcard oder einem verschlüsselten USB-Stick abgelegt werden. Ist eine Speicherung auf der Festplatte unumgänglich, so sind die Daten unbedingt verschlüsselt abzulegen, da ein Zugriffsschutz über das Betriebssystem nur einen unzureichenden Schutz

[12] Den jeweils aktuellen Lagebericht des BSI findet man unter https://www.bsi.bund.de/DE/Publikationen/Lageberichte/lageberichte_node.html

gewährt. Ist ein Angreifer im physischen Besitz einer Festplatte, z.B. durch Diebstahl eines Notebooks, so ist es sehr leicht, das installierte Betriebssystem durch Booten eines unsicheren Betriebssystems zu umgehen und direkt auf den Speicher zuzugreifen. Werden dennoch sensible Daten gespeichert, so sind starke kryptografische Verfahren zu deren Verschlüsselung einzusetzen und der Zugriff auf die verschlüsselten Daten ist zu beschränken. Zugriffsrechte für Quellcode-Daten sollten nur besonders privilegierte Benutzer erhalten, um das gezielte Einschleusen von Trojanischen Pferden zu beschränken.

Zur Abwehr der besprochenen Bedrohungen ist es wichtig, die Rechte von Benutzern zur Ausführung von Betriebssystemdiensten so zu beschränken, dass jeder Benutzer nur diejenigen Dienste nutzen darf, die er zur Erledigung seiner Aufgaben benötigt (Prinzip der minimalen Rechte). Die Basis dafür ist eine Charakterisierung und Klassifizierung von Betriebssystemdiensten, die für manipulatorische Zwecke missbraucht werden können. Beispiele für solche Betriebssystemfunktionen sind:

kritische Dienste

- Befehle zur Änderung von Schutzattributen an Dateien,
- Funktionen zum Lesen und Bearbeiten von Passwortdateien,
- Anweisungen, um Netzverbindungen zu anderen Rechnern zu öffnen, und
- Anweisungen, die einen direkten Zugriff auf Speicherbereiche zulassen.

Modellierung

Beschränkt man die Zugriffsrechte eines Benutzers, so sind auch seine Manipulationsmöglichkeiten entsprechend beschränkt. Um Zugriffsrechte nach dem Prinzip der minimalen Rechte systematisch vergeben zu können, sind genaue Kenntnisse über die Funktionalität der einzelnen Systemdienste erforderlich. Man muss die Aufgabenprofile der einzelnen Benutzer präzise erfassen können und die Zusammenhänge kennen, die zwischen den Systemkomponenten bestehen. Des Weiteren ist es notwendig, die Auswirkungen von Rechtewahrnehmungen zu kennen. Da die betrachteten Systeme, insbesondere Betriebssysteme, komplexe Programme sind, kann man einen entsprechenden Überblick nur dadurch erhalten, dass man das System auf einer geeigneten Abstraktionsebene durch ein Modell beschreibt und die Funktionalität der Systembausteine semi-formal oder formal spezifiziert und verifiziert. Das Modell muss die zu schützenden Objekte und die agierenden Subjekte geeignet erfassen, Festlegungen für die in dem System zu vergebenden Zugriffsrechte treffen sowie die Sicherheitsstrategie des Systems spezifizieren. Auf der Basis des Sicherheitsmodells können Sicherheitseigenschaften überprüft werden. Die bekanntesten und wichtigsten Klassen von Sicherheitsmodellen werden in Kapitel 6 vorgestellt.

Signieren

Weiterhin kann man Programme mit der digitalen Unterschrift des Erzeugers versehen und vor der Programmausführung die Korrektheit der Signatur überprüfen. Die digitale Unterschrift sollte für den eindeutigen, digitalen Fingerabdruck, also einen kryptografischen Hashwert des Programms erstellt werden, so dass auf dieser Basis auch die Unveränderlichkeit des Codes überprüft werden kann (vgl. Kapitel 8). Signierter Code bietet natürlich keine Gewähr vor Trojanischen Pferden. Die Vorgehensweise basiert darauf, dass der Empfänger des signierten Codes dem Signierer vertraut. Das heißt, dass der Empfänger darauf vertraut, dass der Signierer Code ohne Schadsoftware-Anteile erzeugt; eine Verifikation, ob dies tatsächlich der Fall ist, findet beim Validieren der Signatur aber nicht statt. Zur Überprüfung von Signaturen benötigt man ein Zertifikat des Signierers und eine Infrastruktur zur Verteilung der Zertifikate, eine so genannte Public-Key Infrastruktur (vgl. Kapitel 9.1). Die Aussagekraft einer digitalen Signatur hängt in hohem Maß von der Vertrauenswürdigkeit der Zertifikate ab.

Trojanische Pferde werden häufig über Viren, Würmer und Buffer-Overflows in ein System eingeschleust. Dabei wird beispielsweise versucht, ein bereits vorhandenes ausführbares Programm bzw. einen Systemdienst durch modifizierten ausführbaren Code zu ersetzen. Eine andere Möglichkeit des Einschleusens besteht darin, den Quellcode vorhandener Dienste bzw. Programme direkt zu manipulieren und durch eine erneute Übersetzung den ausführbaren Objektcode auf der Maschine zu erzeugen. Auf diese Weise vermeidet der Angreifer, dass sein Code aufgrund von Portierungsproblemen auf dem Zielsystem nicht ausführbar ist. Modifikationen des Quellcodes lassen sich durch Quellcodeüberprüfung bzw. Verifikation aufdecken. Dieser Ansatz ist jedoch bei der Komplexität der heutigen Programme und der mangelnden automatischen Unterstützung mit erheblichem Aufwand verbunden und in der Regel nicht praktikabel. Zur Verhinderung von Quellcodemodifikationen ist deshalb dafür zu sorgen, dass der Code nicht direkt zugreifbar im Speicher liegt, also durch Zugriffsrechte und Verschlüsselungen gesichert wird.

Code-Inspektion

2.6 Bot-Netze und Spam

Bot-Netze und Spam sind in heutigen IT-Systemen nahezu allgegenwärtig. Bot-Netze nutzen gezielt verbreitete Schadsoftware, wie Viren, Würmer und Trojaner, um insbesondere verteilte Angriffe unter Mitwirkung einer großen Anzahl von Rechnern durchzuführen, wobei das Bedrohungspotential dieser Angriffe durch die gezielte Koordinierung tausender befallener Rechner erheblicher größer ist, als durch einen infizierten Rechner. Prognosen, wie der Emerging Cyber Threat Report for 2009 des Georgia Institutes of Technolo-

gy[13] prognostizieren für das Jahr 2009 einen deutlichen weiteren Anstieg von Angriffen durch Bot-Netze. Spam zählt in der Regel nicht direkt zu Schadsoftware, da Spam-Mails meist lediglich unerwünschte Nachrichten enthalten. Da aber zunehmend Trojaner über Massen-Mails wie Spam verbreitet werden, widmet sich ein Unterabschnitt dem Spam-Problem.

2.6.1 Bot-Netze

Bot

Ein Bot-Netz ist ein Verbund von infizierten Rechnern, den so genannten Bot-Rechnern, die miteinander kommunizieren und meist durch einen zentralen Server kontrolliert und ferngesteuert werden. Angreifer missbrauchen hierzu sehr häufig den IRC (Internet Relay Chat, vgl. (RFC 2810)), das Protokolle zum Austausch von Text-Nachrichten in Realzeit, das Chatten, bereit stellt. Das Kunstwort *Bot* ist eine Ableitung von dem Wort Robot, das wiederum vom tschechischen Wort Robota (deutsch *Arbeit*) abstammt. Allgemein ist ein Bot ein Programm, das ohne menschliche Eingriffe Aktionen ausführt. Im Kontext der IT-Sicherheit verstehen wir unter einem Bot ein Programm, das von einem Angreifer gezielt ferngesteuert wird. Bots werden herkömmlicher weise über Schadsoftware wie Viren, Würmer und Trojaner verbreitet. Die in einem Bot-Netz verbundenen Rechner führen ihre Angriffe in der Regel völlig transparent für den Besitzer des jeweiligen Rechners aus.

Angriffe

Bot-Netze können aus mehreren tausend Rechnern bestehen. Hauptangriffsziele sind verteilte Denial-of-Service Angriffe (DDoS) auf Anbieter von Internetdiensten und das massenhafte, über verteilte Absender (die Bots) initiierte Versenden von Spam-Nachrichten. Durch die ferngesteuerte Kontrolle von tausenden manipulierter Rechner steht einem Angreifer eine Bandbreite zur Durchführung eines Angriffs zur Verfügung, die die der meisten herkömmlichen Internetzugänge um Größenordnungen übertrifft. Mit einem ausreichend großem Bot-Netz ist es somit möglich, über das Versenden von großen Datenmengen attackierte Serviceanbieter zu überlasten; also einen klassischen Denial-of-Service herbei zu führen. Neben diesen Angriffen, für die die Bot-Rechner als Zombies instrumentalisiert werden, können Bot-Programme natürlich auch Angriffe auf die befallenen Bot-Rechner selber durchführen und analog zu klassischen Viren und Trojanern als Key-logger agieren, Netzwerkverbindungen zum Download von Malware aufbauen oder aber auch Viren-Scanner auf dem befallenen Rechner deaktivieren.

IRC

Wie bereits erwähnt, wird nach wie vor zur zentral koordinierten Fernsteuerung eines Bot-Netzes zumeist das IRC verwendet. Abbildung 2.7 veranschaulicht die allgemeine Struktur eines derartigen Bot-Netzes. Nach

[13] vgl. http://www.gtisc.gatech.edu/pdf/CyberThreatsReport2009.pdf

der Infizierung werden die Bot-Programme aktiviert und verbinden sich mit einem IRC-Server. Der IRC-Server dient als Relaisstation (daher Internet Relay Chat - IRC), damit die Bot-Programme miteinander über einen gemeinsamen Kanal, dem sie explizit beitreten, kommunizieren können.

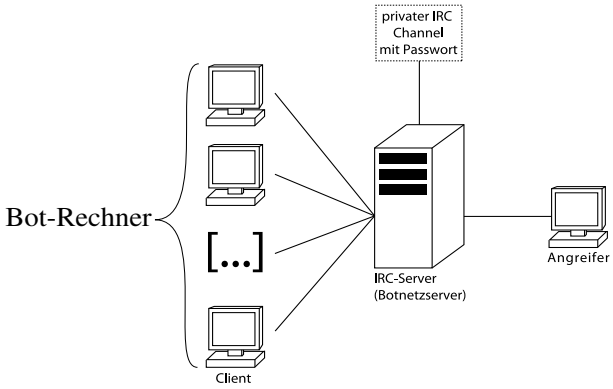

Abbildung 2.7: IRC-kontrolliertes Bot-Netz

Um die Mitgliedschaft in einem Bot-Netz zu kontrollieren, ist ein solcher Kanal zumeist Passwort-geschützt und das jeweilige Passwort wird nur den Bot-Programmen bekannt gegeben. Die Fernsteuerung der Bot-Rechner erfolgt dann durch einen Administrator des IRC-Servers. Beispiele für Kommandos, die ferngesteuert an die Bots weitergeleitet werden, sind die Aufforderung, nach weiteren Systemen, die infiziert werden können, zu scannen, einen DoS-Angriff auszuführen oder aber auch einen Download durchzuführen und das geladene Programm anschließend zu starten.

Abwehr

Da der IRC-Server bei diesen Angriffen die zentrale Kontrolleinheit ist, kann man das Bot-Netz deaktivieren, wenn es gelingt, diesen IRC-Server aus dem Netz zu entfernen. Um zu erkennen, ob ein IRC-Server ein Bot-Netz kontrolliert, wird in der Regel der Datenverkehr überwacht und auf verdächtige Nachrichten hin gescannt. Da zur Fernsteuerung von Bot-Rechnern der IRC-Server, wie oben beschrieben, Steuerbefehle zur Initiierung von DDoS-Angriffen, zum Download von Malware bzw. zur Durchführung von Scans übermittelt, kann mittels eines Netzwerk Intrusion Detection Systems gezielt nach den Signaturen derartiger Steuerbefehle, also deren syntaktischer Struktur, gesucht werden. Da die Kommunikation in einem IRC-basierten Bot-Netz im Klartext erfolgt, ist eine derartige Überwachung möglich.

Trends

Die nächste Generation der Bot-Netze bedient sich jedoch zunehmend dezentraler Peer-to-Peer (P2P)-Strukturen und verwendet eine verschlüsselte Kommunikation. Dadurch wird sowohl das Erkennen eines Bot-Netzes als auch dessen Deaktivierung erheblich erschwert, da es ja gerade ein Charak-

teristikum eines dezentralen P2P-Netzes ist, dass es auch nach dem Ausfall eines Peers seine Funktionalität weiter aufrecht erhalten kann und sich selbst-organisierend rekonfiguriert.
Ein weiterer Trend, der auch vom BSI beobachtet wird, ist die Zunahme an Bot-Netzen, die vermehrt zu gezielten, kriminellen Aktivitäten eingesetzt werden. So können Bot-Netze von Kriminellen gemietet werden, um gezielt zum Beispiel einen Service-Provider mittels eines verteilten Denial-of-Service Angriffs zu blockieren und erst nach Zahlung eines Geldbetrages diese Blockade wieder aufzuheben.

2.6.2 Spam

Spam

Neben Viren, Würmern und Trojanern, die erheblichen finanziellen Schaden in heutigen Systemen bewirken können, zählt die zunehmende Flut der Spam-Mails zu einem großen Übel, das Unternehmen erhebliche Ressourcen kostet, um ihrer Herr zu werden. Unter Spam versteht man eine Massen-E-Mail, häufig Werbesendungen, die im Internet verbreitet wird. Diese Werbung wird unaufgefordert an Millionen von E-Mail-Adressen versendet. Die Adressen stammen von Adresshändlern, oder werden durch spezielle Programme im Internet gesucht. Ursprünglich handelt es sich bei Spam um Pressfleisch in Dosen, Spiced Porc And Meat. Der Begriff Spam-Mail geht angeblich auf einen Sketch von Monty Python zurück, in dem ein Restaurantgast nur Gerichte bestellen konnte, wenn er gleichzeitig auch Spam geordert hat. In Übertragung auf das Internet bedeutet das, dass ein Benutzer die Vorzüge des Internets und insbesondere des sehr kostengünstigen Versendens von E-Mails nur nutzen kann, wenn auch bereit ist, Spam-Müll in Kauf zu nehmen.

Schäden

Untersuchungen z.B. von Nucleus Research (USA) gehen davon aus, dass jeder Mitarbeiter in einem Unternehmen im Durchschnitt täglich 13,3 Spam-Mails erhält, für die er 6,5 Minuten benötigt, um diese zu lesen und danach zu löschen. Das summiert sich jährlich auf etwa 25 Arbeitsstunden pro Kopf und macht etwa 1,4 Prozent der gesamten Arbeitszeit aus. Insgesamt ergeben sich durch die Spam-Mails erhebliche Verluste an Produktivität, die zu sehr hohen gesamtwirtschaftlichen Kosten führen. So kommt 2004 das US-Forschungsinstitut Ferris Research in einer Studie über Spam-Mails zu dem Schluss, dass allein in den USA Einbußen bis zu zehn Milliarden Dollar jährlich aufgrund spamverseuchter E-Mails anfallen. Neben den Kosten, die direkt durch diese Mails verursacht werden, müssen auch die Kosten betrachtet werden, die für die Entwicklung und Aufrechterhaltung von Antispamsystemen aufzubringen sind. Ferris Research schätzt, dass US-Unternehmen allein im Jahr 2004 dafür bis zu 120 Millionen Dollar aufbringen mussten.

Abwehr

Zur Abwehr der Spam-Flut werden Spam-Filterungen eingesetzt. Zur Filterung von Spam stehen im Internet frei verfügbare Filterprogramme zur Verfügung, wie beispielsweise SpamProbe (vgl. http://spamprobe.sourceforge.net/) für Unix-Derivate oder auch für Windows-Systeme unter der Cygwin-Umgebung. In Unternehmen erfolgt die Spam-Filterung häufig über zentral administrierte Mailrelays, die neben einer Spam-Filterung auch in der Regel eine Virenerkennung durchführen. Neben der Spam-Erkennung durch Spam-Filterungsprogramme muss über Regelwerke festgelegt werden, wie mit Mails, die unter Spam-Verdacht stehen, umgegangen wird. Häufig beschränken sich die Filter darauf, in der Subject-Zeile der Mail diese als mögliche Spam zu markieren und ggf. nicht an den Empfänger weiterzuleiten, sondern sie in einer Spam-Quarantäne-Datei für den Empfänger zu sammeln und regelmäßig, z.B. einmal wöchentlich, dem Empfänger eine Mail zu schicken, in der die Subject-Zeilen und Absenderadressen aller als Spam eingestufter Mails zusammengefasst werden. Manche Unternehmen gehen jedoch auch soweit, Spam-Mails direkt zu löschen.

Rechtslage

Beide Vorgehen, also das Ändern der Subject-Zeile und das Löschen sind jedoch rechtlich bedenklich. Gemäß §303a des Strafgesetzbuches (StGB) macht sich jemand strafbar, der rechtswidrig Daten löscht, unterdrückt, unbrauchbar macht oder verändert. Ob die Subject-Zeile als geschützter Bestandteil der Mail anzusehen ist, wird in juristischen Kreisen noch stark diskutiert. Die DFN Forschungsstelle Recht in Münster rät davon ab, Subject-Zeilen zu manipulieren. Das Löschen von Spam ist laut dieser Forschungsstelle ebenfalls rechtlich nicht unbedenklich, falls in Unternehmen die private Nutzung der E-Mails zugelassen oder geduldet wird. Das Fernmeldegeheimnis ist zu beachten, und gemäß §206 Abs. 2 Nr. 2 des Strafgesetzbuches müssen grundsätzlich alle Mails ohne Verzögerung zugestellt werden. Ein Nutzer kann jedoch sein Einverständnis zum Löschen von Spam erklären.
Das Filtern von Spam-Mails ist aber auch problematisch, weil dadurch in das Fernmeldegeheimnis eingegriffen wird, das nicht nur den Inhalt, sondern auch die Umstände der Telekommunikation schützt. Nach §206 StGB kann derjenige mit einer Freiheitsstrafe bis zu fünf Jahren bedroht werden, der „unbefugt einer anderen Person eine Mitteilung über Tatsachen macht, die dem Post- und Fernmeldegeheimnis unterliegen und die ihm als Inhaber oder Beschäftigtem eines Unternehmens bekannt geworden sind, das geschäftsmäßig Post- oder Telekommunikationsdienste erbringt.“Auch hier muss also für das Filtern die Einwilligung des E-Mail Empfängers eingeholt werden.

Fazit

Nach wie vor steigen die Angriffe durch Schadsoftware weiter an. Im Gegensatz zu früher ist hierbei jedoch der Anteil an Viren und Würmer

erheblich zurückgegangen und der Anteil an Trojanischen Pferden liegt bei über 70 Prozent. E-Mails sind noch immer häufig genutzte Verbreitungswege für Schadsoftware. Als Ausgangspunkt für die Verbreitung dienen aber zunehmend URLs und gefälschte Web-Seiten, die vermeintlich interessante Downloads anbieten, so dass sich Benutzer sehr häufig auf diesem Weg einen Trojaner auf ihrem System installieren. Spam-Mails oder aber Massen-Emails mit verseuchtem Anhang könnten durch die konsequente Nutzung von Signaturen eingedämmt werden. Es ist kritisch zu fragen, ob hier nicht auch die Politik und die Gesetzgebung gefordert sind, um über Regelungen zur Eindämmung der Flut nachzudenken.

2.7 Mobiler Code

Mit der Zunahme des verteilten Berechnens u.a. durch Agentensysteme und insbesondere mit der Verbreitung der plattformunabhängigen Programmiersprache Java wächst der Einsatz von so genanntem mobilen Code. Im Zusammenhang mit Smartphones sind mobile Apps in den letzten Jahren äußerst populär geworden. Die Frage der Sicherheit tritt dabei verstärkt zu Tage, da vermehrt Apps auf mobilen Endgeräten in Umlauf gebracht werden, die Schadcode mit sich führen. Wir gehen in diesem Abschnitt deshalb abschließend auf Sicherheitsprobleme speziell im Zusammenhang mit mobilen Endgeräten und Apps ein.

2.7.1 Eigenschaften

Im Gegensatz zu den in diesem Kapitel bereits eingeführten Viren, Würmern und Trojanischen Pferden wird mobiler Code nicht vordringlich zum Zwecke eines Angriffs auf einen entfernten Rechner eingesetzt. Das Konzept enthält aber ein großes Bedrohungspotential, weswegen an dieser Stelle auf allgemeine Sicherheitsprobleme im Zusammenhang mit mobilem Code eingegangen wird.

Definition 2.4 (Mobiler Code, aktiver Inhalt)

mobiler Code

Unter dem Begriff des mobilen Codes oder des aktiven Inhalts lassen sich alle Programme zusammenfassen, deren Code auf einem entfernten, potentiell nicht vertrauenswürdigen Rechner generiert wurde und die auf Gastrechnern (engl. *host*) ausgeführt werden.

□

Mobiler Code, oder ein mobiler Agent als spezielle Ausprägung davon, wandert von einem Rechnerknoten in einem Netz zu einem anderen und wird dort ausgeführt. Damit unterscheidet sich das zugrunde liegende Ausführungsmodell grundlegend vom herkömmlichen Client-Server Mo-

dell zur Durchführung verteilter Berechnungen. Beim Client-Server Modell werden Nachrichten, also Daten, als Eingabe- und Ausgabeparameter für Dienstaufrufe (z.B. Remote Procedure Call (RPC) oder Remote Method Invocation (RMI)) über ein unsicheres Transportmedium übertragen. Der ausführbare Code selbst, also der angebotene Dienst, bleibt jedoch permanent auf dem Rechner, auf dem er installiert wurde.

Beispiele

Bekannteste Beispiele für mobilen Code bzw. aktive Inhalte sind Java-Applets [54], ActiveX-Controls von Microsoft, dynamische Mails mit Kommandos einer Skriptsprache (u.a. Tcl) [31] oder auch mobile Agenten. Ein spezielles Tag in einer HTML-Seite gibt an, ob es sich bei dem mobilen Code um ein Java Applet (angezeigt durch das Tag < Applet >), um JavaScript-Code (Tag < Script Language = JavaScript >) oder beispielsweise um ein ActiveX-Control (Tag < Object >) handelt.

2.7.2 Sicherheitsbedrohungen

Bedrohungen

Im Zusammenhang mit dem Einsatz von mobilem Code treten im Wesentlichen drei Klassen von Sicherheitsbedrohungen auf, die sowohl den Code selbst als auch die Ausführungsumgebung seines Gastrechners betreffen. Mobiler Code wird über ein Transportmedium übertragen, so dass er auf diesem Transportweg vielfältigen Bedrohungen ausgesetzt ist. Wird er auf einem Gastrechner zur Ausführung gebracht, so ist der mobile Code Bedrohungen seitens dieses Rechners ausgesetzt. Natürlich stellt der mobile Code auch seinerseits eine Bedrohung für seinen Gastrechner dar. Dieser muss sicherstellen können, dass bei der Ausführung des Codes keine unautorisierten Zugriffe erfolgen, Ressourcen nicht unzulässig beansprucht oder Viren und Trojanische Pferde auf den Gastrechner übertragen werden. Die drei Bedrohungsbereiche sind in Abbildung 2.8 skizziert und werden im Folgenden genauer untersucht.

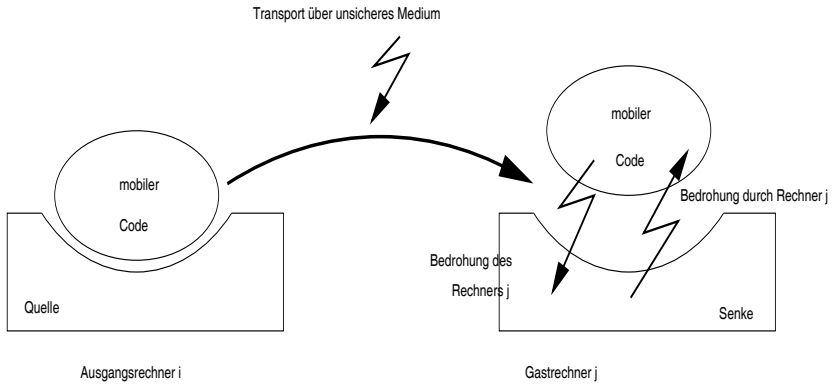

Abbildung 2.8: Bedrohungsszenarien für mobilen Code

Bedrohungen des mobilen Codes

passive Angriffe

Passiven Angriffen ist der mobile Code sowohl beim Transport über ein unsicheres Medium als auch bei seiner Ausführung ausgesetzt. Aus der Sicht des Codes ist sicherzustellen, dass er nur von berechtigten Subjekten ausgeführt wird und dass bei der Ausführung seine lokalen, sensiblen Daten nicht unautorisiert an unberechtigte Dritte weitergegeben werden. Beispielsweise könnte bei einer interpretativen Ausführung des mobilen Codes auf dem Gastrechner der Interpreter der Ausführungsumgebung Zugriff auf lokale Kommunikationsschlüssel erhalten und diese zu Missbrauchszwecken speichern. Falls der Gastrechner von einem externen Angreifer kontrolliert wird, so kann diese Information auch direkt an den Angreifer weitergeleitet werden.

aktive Angriffe

Durch aktive Angriffe kann der Code beim Transfer oder im Verlauf der Ausführung modifiziert, oder die Ausführung des Codes kann absichtlich vorzeitig beendet werden. Dies entspricht einem Denial-of-Service (DoS) Angriff. Auf diese Weise kann zum Beispiel ein Virus oder ein Trojanisches Pferd integriert oder bei mobilen Agenten der vorgesehene Ausführungspfad verändert werden, so dass der Agent gezielt auf einen Rechner umgelenkt oder von einem vorgesehenen Gastrechner absichtlich ferngehalten wird. Bedrohungen durch Manipulationsangriffe auf den mobilen Code ergeben sich insbesondere auch im Kontext von Electronic Commerce. Hat z.B. ein Agent die Aufgabe, Preise für ein Produkt aufzusammeln, so kann ein Gastrechner die bereits gespeicherten Preise von Konkurrenten löschen oder modifizieren bzw. verhindern, dass der Code den Rechner eines preisgünstigeren Kontrahenten besucht.

Bedrohungen für den Gastrechner

Mobiler Code stellt eine erhebliche Bedrohung für seinen Gastrechner dar, falls keine Maßnahmen ergriffen werden, den Zugriff auf die lokalen

passive Angriffe

Ressourcen des Gastsystems zu beschränken. Durch passive Angriffe wie den lesenden Zugriff auf Dateien können Informationen gewonnen und über eine Netzwerkverbindung an den Quellrechner des Codes gesendet werden. Ein bekanntes Beispiel hierfür ist das Lesen bzw. Transferieren der Passwortdatei des Gastrechners zum Rechner des Angreifers, so dass dort gezielt ein Passwort-Crackangriff durchführbar ist. Passive Angriffe können auch das Abhören der Kommunikation zwischen anderen mobilen Agenten, die auf dem System ausgeführt werden, betreffen. Auf diese Weise lassen sich Informationen über deren Aktivitäten sammeln und für einen Angriff ausnutzen.

aktive Angriffe

Mittels aktiver Angriffe kann der mobile Code versuchen, einen unautorisierten lesenden oder modifizierenden Zugriff auf die Dateien oder direkt auf die

Speicherbereiche seines Gastsystems zu erlangen. Aktive Angriffe betreffen auch den Bereich des Denial-of-Service. Die Ausführung von mobilem Code kann dazu führen, dass die CPU monopolisiert wird, so dass andere Anwendungen blockiert werden. Weitere Ausprägungen von Angriffen auf die Verfügbarkeit von Ressourcen betreffen das „Lähmen“ des Gastrechners durch das Erzeugen einer Vielzahl von geöffneten Fenstern oder durch das Ausschöpfen der vorhandenen Speicherkapazität, so dass das System nur noch mit Verwaltungsmaßnahmen beschäftigt ist (so genanntes Thrashing) und die Berechnungen der Anwendungen nicht fortschreiten.

Maskierung

Probleme ergeben sich weiterhin aus Maskierungsangriffen seitens des mobilen Codes. Dabei ist zu unterscheiden zwischen der Vortäuschung einer falschen Identität und der Vortäuschung einer falschen Funktionalität. Zu beachten ist auch bei mobilem Code, dass Maßnahmen wie das Signieren von Code keine Auskunft über die Funktionalität geben. Auch signierter Code kann eine bedrohliche Funktionalität beinhalten!

Beispiel 2.6 (ActiveX Control)

ActiveX

ActiveX (u.a. [83]) ist eine von Microsoft entwickelte Erweiterung der OLE (Object Linking and Embedding) Technologie zur Wiederverwendung von Komponenten. ActiveX ist Bestandteil der Microsoft Betriebssysteme Windows 95/98/2000/XP und NT. Ein ActiveX Control ist eine wiederverwendbare Komponente in binärer Form mit einer standardisierten Schnittstelle. Es kann in einer beliebigen Programmiersprache wie C oder Basic erstellt sein. ActiveX Controls sind nicht eigenständig ablauffähig, sondern benötigen einen Wirt. Dies kann beispielsweise ein Browser sein. Soll dieser beim Anzeigen einer HTML-Seite ein ActiveX Control laden, so wird dies durch das Tag $<$ OBJECT ID = ... $>$ in der Seite angezeigt. Der ebenfalls auf der Seite enthaltene Eintrag CODEBASE gibt an, wo sich der Code des Controls befindet. Der Browser ruft nun den Internet Component Download Service (ICDS) auf, der das Control anhand der angegebenen Adresse aus dem Internet lädt. Das Control wird innerhalb des Browsers ausgeführt, der ein Bestandteil des Microsoft-Betriebssystems ist und entsprechend viele Zugriffsrechte besitzt. Damit erhält auch ein Control einen unbeschränkten Zugriff auf alle Systemressourcen, also auch auf das Dateisystem!

▲

2.7.3 Gegenmaßnahmen

Verschlüsseln

Durch den Einsatz von Verschlüsselungstechniken kann auf relativ einfache Weise der Transport mobilen Codes abgesichert werden, jedoch müssen für eine interpretative Ausführung des Codes die benötigten Daten der Ausführungsumgebung unverschlüsselt vorliegen. Der Schutz dieses Pro-

grammcodes ist somit durch kryptografische Verfahren nicht zu gewährleisten. Eine Lösungsmöglichkeit besteht darin, mobilen Code nur zu solchen Systemen zu versenden, die vertrauenswürdig sind und die ihre Identität korrekt nachgewiesen haben (vgl. Kapitel 10). Mit den Möglichkeiten, die das Trusted Computing auf der Basis des Trusted Platform Modules (TPM) eröffnet (vgl. Abschnitte 11.3), stehen nunmehr auch weitergehende Mechanismen zur Verfügung, um die Vertrauenswürdigkeit von potentiellen Gastrechnern zu prüfen. Über das so genannte Remote-Attestation Protokoll, das ein TPM-Chip unterstützt (siehe Seite 622), ist es möglich, einen Gastrechner nach seiner Systemkonfiguration zu befragen. Da dies jedoch nur ein rudimentärer Mechanismus dafür ist, Vertrauen in die unmanipulierte Arbeitsweise eines Gastrechners aufzubauen, kann man auch vorab auf Gastrechnern spezielle Ausführungsumgebungen installieren, in denen dann der mobile Code zur Ausführung gebracht wird. Diese Vorgehensweise ist aus der Welt der Agentensysteme oder aber auch der DRM-Systeme (Digital-Rights-Management) bekannt.

vertrauenswürdige Gastrechner

In Forschungsarbeiten an der TU München[14] und am Fraunhofer AISEC[15] wird diese Idee mit Virtualisierungskonzepten und den Möglichkeiten, die der TPM bietet, gekoppelt, um eine vertrauenswürdige Umgebung zur Durchführung von e-Commerce Transaktionen, bei denen auch mobiler Code auf den Gastrechner, das ist hier der PC des Kunden, heruntergeladen werden kann, zu schaffen (vgl. u.a. [202, 201]). Dies liefert erste Ansätze zum Schutz mobilen Codes bei dessen Ausführung auf einem unbekannten Gastrechner. Weitere Forschungsarbeiten sind hier jedoch erforderlich, um zum einen die Flexibilität der Lösungen zu verbessern und zum anderen die Qualität der Attestierungsaussagen zu erhöhen, da zurzeit lediglich Konfigurationsüberprüfungen stattfinden. Wünschenswert wäre es, Aussagen über das korrekte Verhalten und andere Eigenschaften des Gastrechners vorab prüfen zu können.

Schutz des Gastrechners

Wie schon bei den Software-Anomalien, so ist auch bei der Abwehr von Bedrohungen durch mobilen Code eine Beschränkung der Zugriffsrechte des ausführbaren Codes unumgänglich. Eine sehr restriktive Vorgehensweise ist die so genannte Sandbox, wie sie von Safe Tcl und Java JDK 1.1 realisiert wird. Die Sandbox definiert die Ausführungsumgebung des Codes als einen isolierten und isolierenden Speicherbereich, innerhalb dessen der Code freien Zugriff besitzt, aber Zugriffe auf Objekte außerhalb des Bereichs kaum oder nur unter sehr strengen Kontrollen möglich sind. Die Nutzung von

Sandbox

[14] http://www.sec.in.tum.de/
[15] http://www.aisec.fraunhofer.de

Virtualisierungstechniken (u.a. [202]) stellen eine Verallgemeinerung der Sandbox-Technik dar und ermöglichen es, Subsysteme, die beispielsweise nur wenige Komponenten umfassen, in einer virtuellen Maschine ablaufen zu lassen, und diesen Bereich von anderen sensitiven Bereichen des Gastsystems zu isolieren.

Zur Abwehr von Maskierungsangriffen muss der Code eindeutig identifizierbar sein. Dazu werden in der Praxis vielfach Authentizitätsnachweise eingesetzt. Hierunter fällt auch die so genannte Authenticode-Technik (vgl. Seite 422), die von Microsoft als Schutz vor bedrohlichen ActiveX-Controls empfohlen und eingesetzt wird. Diese Technik bietet jedoch nur einen sehr schwachen Schutz, da lediglich der Ursprung einer mobilen Einheit nachgewiesen und überprüft werden kann. Die Überprüfung basiert darüber hinaus auf Zertifikaten, die man sich für geringes Entgelt ausstellen lassen kann. Ein signiertes und zertifiziertes ActiveX-Control oder Java-Applet erlaubt jedoch keinerlei Aussage über die Vertrauenswürdigkeit des Codes, der ausgeführt wird. Er kann Viren und Trojanische Pferde enthalten und eine absichtliche oder unabsichtliche Bedrohung des Systems darstellen. Ein Signieren von mobilem Code als einzige Abwehrmaßnahme ist somit völlig unzureichend und eher kontraproduktiv, da er Benutzer in einer falschen Sicherheit wiegt.

Authenticode

Nachweise, dass der mobile Code eine spezifizierte Funktionalität und nur diese erfüllt, sind sehr viel schwieriger zu erstellen und nachzuprüfen als Ursprungsnachweise. Mit der Technik des Proof-Carrying Codes [150] wurde ein interessanter Ansatz entwickelt. Die grundlegende Idee dabei ist, dass ein Gastrechner zunächst eine Sicherheitsstrategie festlegt und veröffentlicht. Der ausführbare, mobile Code wird vom Erzeuger mit einem Beweis ausgestattet, der nachweist, dass der Code die spezifizierte Strategie erfüllt. Vor der Ausführung dieses Codes kann dann der Gastrechner den Beweis überprüfen und damit ohne aufwändige, zur Laufzeit durchzuführende Kontrollen sicherstellen, dass bei der Ausführung des mobilen Codes keine Strategie-Verletzung auftreten wird. Zu den Eigenschaften, die auf diese Weise nachweisbar sind, gehören Aussagen darüber, dass der Code nur Speicheradressen eines bestimmten Adressbereichs verwendet oder dass keine Pufferüberläufe auftreten können, da die Einhaltung von Bereichsgrenzen kontrolliert wird. Die Erstellung von Beweisen ist jedoch keinesfalls eine triviale Aufgabe und nur in Ausnahmefällen automatisch durchführbar. Deshalb wird diese Technik wohl erst dann auf eine breite Akzeptanz stoßen, wenn einfach handhabbare Entwicklungsumgebungen zur Konstruktion solcher erweiterter Code-Bausteine zur Verfügung stehen.

Proof-Carrying Code

2.7.4 Mobile Apps

Apps

Mobile Apps[16] sind bereits heute im Consumer-Bereich sehr weit verbreitet. Derzeit wird noch ein großer Teil der Apps als rein lokale Anwendungen auf mobilen Geräten ausgeführt, jedoch ist die Entwicklung hin zu smarten mobilen Apps deutlich zu sehen. Kennzeichnend für diese fortgeschrittenen Apps ist ihre Eigenschaft, Dienste zu nutzen, die über das Internet oder ein privates Netzwerk bereitgestellt werden. Viele dieser Dienste werden bereits heute in einer Cloud betrieben und ermöglichen es dem Nutzer somit, einen konsistenten Datenbestand auf unterschiedlichen mobilen und stationären Geräten zu führen. Eine App stellt dabei die Client-Seite eines Cloud-Dienstes dar, so dass es für den Nutzer keine Rolle mehr spielt, von wo und mit welchem Gerät der Zugriff auf seine Daten erfolgt. Mit diesen Eigenschaften werden smarte Apps zunehmend auch für den Geschäftsbereich attraktiv. Als Business-Apps werden sie ein integraler Bestandteil von Geschäftsprozessen werden.

Sicherheitsbedrohungen

Smartphones und Tablet-PCs werden von Nutzern als eine Art persönliche Informationszentralen verwendet. Die Geräte sind *always on* und werden von vielen Benutzern stets mitgeführt. Von zentraler Bedeutung für die Sicherheit der Anwendungen, und damit natürlich auch der Apps, ist die Systemsicherheit der mobilen Plattform, auf der die Anwendungen zur Ausführung gelangen. Der zweite sicherheitskritische Bereich wird durch die Anwendungsebene definiert. Sie bildet die direkte Ausführungsumgebung für die Apps.

Bedrohungen der Mobilen Plattformen

Plattform

Mobile Plattformen unterliegen einer höheren Gefährdung als stationäre. Sie werden an verschiedenen Orten und damit in unterschiedlich vertrauenswürdigen Umgebungen eingesetzt. Sie gehen schneller verloren oder werden gestohlen als etwa ein PC. Gleichzeitig kann sich ein Unbefugter Zugang zu einem mobilen Gerät verschaffen. Die kritischste Bedrohung für mobile Plattformen ist jedoch, dass solche personalisierten Geräte oftmals nicht oder nur unzureichend von einer unternehmensweiten Sicherheitsstrategie erfasst werden. Meistens werden sie von den Benutzern selbst administriert. Die Erfahrung aus der Welt der Desktop-Betriebssysteme lehrt jedoch, dass noch immer vielen Anwendern ein hinreichendes Sicherheitsbewusstsein fehlt: Zugriffssperren und Passwortschutz werden deaktiviert, da sie als störend empfunden werden, Sicherheitswarnungen werden ignoriert,

[16] Der Abschnitt basiert auf einem Teil eines Buchkapitels, das ich zusammen mit Christian Schneider verfasst habe. Das Buch ist unter dem Titel *Smart Mobile Apps* im Springer Verlag erschienen.

Software-Updates werden, wenn überhaupt, sehr unregelmäßig eingespielt. In solchen Fällen vergrößert sich entsprechend das bereits vorhandene Bedrohungspotential mit einer unnötig großen Angriffsfläche der mobilen Plattform.

Update-Problematik

Viele mobile Betriebssysteme müssen für das Einspielen von Updates mit einem Rechner verbunden werden. Sogenannte *Over-the-Air* (OTA)-Updates, also Firmware-Updates ohne PC-Verbindung, sind bisher nur für Android, Windows Phone 7 und Symbian 3 verfügbar, unter gewissen Voraussetzungen auch für BlackBerry OS. Ältere Windows Mobile und Symbian-Versionen sowie Apples iOS bieten bisher nur Betriebssystem-Updates mit Unterstützung eines PCs. Googles Android-Plattform, bietet zwar OTA-Updates, leidet aber unter einem anderen Problem: Mit den schnellen Release-Zyklen von neuen Android-Versionen durch Google können viele Hersteller von mobilen Geräten nicht mithalten. Sie stellen deshalb nur für einen kurzen Zeitraum Firmware-Updates für ihre Produkte zur Verfügung, und das auch nur mit deutlicher Verzögerung zur Android-Version[17]. Erschwerend kommt hinzu, dass die Mobilfunkbetreiber viele Smartphones mit speziellen Anpassungen vertreiben. Diese benötigen jedoch ihrerseits speziell angepasst Firmware-Updates, die in der Regel wiederum noch später als die Updates der Hardware-Hersteller zur Verfügung gestellt werden. Die Konsequenz ist, dass auf den meisten Android-Geräten veralteter System-Software installiert ist. Beispielsweise wurde Android 2.3 im Dezember 2010 veröffentlicht, doch nach Angaben von Google war ein halbes Jahr später auf 64,6 Prozent aller Geräte noch Version 2.2 installiert. Im Mai 2011 liefen insgesamt nur 8,1 bzw. 0,3 Prozent aller Android-Geräte mit der aktuellen Version 2.3.4 für die Smartphone-Klasse beziehungsweise Version 3.1 für die Tablet-Klasse. Das heißt, über 90 Prozent aller Android-Geräte verwenden eine veraltete Betriebssystemversion, die möglicherweise bekannte aber noch nicht geschlossene Sicherheitslücken enthält.

Bedrohungen der Anwendungsebene

Malware-App

Die meisten Apps werden über einen speziellen App-Marktplatz bekannt gemacht und vertrieben. Für die mobilen Endgeräte von Apple (iPhone, iPad und iPod) ist Apple's AppStore sogar der einzige Weg, um neue Anwendungen zu beziehen und auf den Geräten zu installieren. Der Nutzer muss also der Qualitätskontrolle der Marktplatzbetreiber vertrauen. D.h. der Nutzer muss darauf vertrauen, dass die über den Marktplatz angebotenen Apps keine Sicherheitslücken oder Schad-Funktionen aufweisen. Da die entsprechenden Kontrollverfahren nicht offengelegt sind und Apps keine standardisierte Zertifizierungsprozedur durchlaufen müssen, kann über die

[17] vgl. Namestnikov, Yury: Malware-Report,2011. Kaspersky Lab, http://www.viruslist.com/de/